DESIGN AND DEVELOPMENT OF A GENERIC ARCHITECTURE FOR APPAREL MANUFACTURING :

Volume V, Research Methodology

Research Sponsored by:

U.S. Defense Logistics Agency
DLA-MMPRT
8725 John J. Kingman Road, Suite 2533
Ft. Belvoir, Virginia 22060-6221

DLA Contract #: DLA900-87-D-0018/0001

Reported by:

Dr. Sundaresan Jayaraman
Principal Investigator

Georgia Tech Project #: E-27-628

Georgia Institute of Technology
School of Textile & Fiber Engineering
Atlanta, Georgia 30332

Tel: 404/894-2490
Fax: 404/894-8780

March 1996

SJ-TR-ARCH-9603A

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	8. REPORT TYPE AND DATES COVERED
	March 14, 1996	Final Project Report: July 11, 1988 - Dec 14, 1995

4. TITLE AND SUBTITLE
Design and Development of a Generic Architecture for Apparel Manufacturing : Volume V, Research Methodology

5. FUNDING NUMBERS

6. AUTHORS(S)

Dr. Sundaresan Jayaraman
Rajeev Malhotra

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Georgia Institute of Technology
School of Textile & Fiber Engineering
Atlanta, Georgia 30332-0295

Through: The Georgia Tech Research Corporation

13. PERFORMING ORGANIZATION REPORT NUMBER
SJ-TR-ARCH-9603A, Volume V
Part Six of Seven-Part Series of Reports

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
US Defense Logistics Agency, DLA-MMPRT
8725 John J. Kingman Road, Suite 2533
Ft. Belvoir, Virginia 22060-6221

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES COR:

12a. DISTRIBUTION/AVAILABILITY STATEMENT

UNLIMITED

12b. DISTRIBUTION CODE

A

13. ABSTRACT *(Maximum 200 words)*

Research has been carried out to design and develop a generic architecture for an apparel enterprise that can serve as a blueprint for a computer-integrated apparel enterprise (CIAE). The Apparel Manufacturing Architecture (AMA) -- the first comprehensive architecture for manufacturing -- has been developed and validated in close collaboration with the apparel industry. AMA consists of a set of models the core of which is the *information* model which defines the schema of the shared information base for an apparel enterprise. The *function* model component of the architecture specifies how the activities carried out in an apparel manufacturing enterprise interact with each other through the shared information base. The third component of AMA, the *dynamics* model, describes how the interactions among the enterprise activities take place over time. The Recruit Induction Center Architecture (RICA) models the uniform distribution process at the Recruit Induction Center (RIC).

Volume V documents research methodology.

19. SUBJECT TERMS
Apparel Manufacturing; Enterprise Architecture; Information Architecture; Computer-Integrated Manufacturing; Modeling; Information Systems; Integrated Databases;

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unclassified / UL

Research Project Personnel

Harinarayanan Balakrishnan
Aruna Cidambi
Rajeev Malhotra
Annajee Rao Nott
Rangaswamy Rajamanickam
M. C. Ramesh
K. Srinivasan
Yin Zhou
Graduate Research Assistants

Dr. Sundaresan Jayaraman
Principal Investigator

Table of Contents

Acknowledgments

This research project was funded by the US Defense Logistics Agency (DLA) under contract number DLA-900-87-0018/0001. The authors would like to thank Mr. Donald F. O'Brien, Mr. Dan Gearing, Ms. Julie Tsao and Ms. Helen Kerlin of DLA for making this research endeavor possible.

Oxford Slacks, Monroe, Georgia, served as the first industry partner on the effort and contributed significantly to the development of the initial version of the Apparel Manufacturing Architecture. The various individuals at Oxford Slacks -- especially Ms. Melissa Bailey, Mr. Wilbur Holloman and Mr. Larry Johnson -- deserve sincere thanks and appreciation for their participation in the research.

Subsequently, the American Apparel Manufacturers Association (AAMA) and several member companies of AAMA participated in evaluating and validating the architecture and deserve thanks.

Finally, Mr. John Adams and Ms. Susan Shows of AMTC provided the necessary administrative support during the project and their efforts are thankfully acknowledged.

* * *

Executive Summary

The Final Technical Report for the project entitled "Design and Development of a Generic Architecture for Apparel Manufacturing" is being submitted in seven volumes. The scope of the individual volumes is as follows:

Volume 0 — Executive Summary Technical Report
[SJ-TR-ARCH-9603]

Volume I — AMA Primer
[SJ-TR-ARCH-9412]

Volume II — Apparel Manufacturing Architecture: The Function Model
[SJ-TR-ARCH-9412]

Volume III — Apparel Manufacturing Architecture: The Information Model
[SJ-TR-ARCH-9412]

Volume IV — Recruit Induction Center Architecture: Function and Information Models for the Uniform Distribution Process
[SJ-TR-ARCH-9411]

Volume V — Research Methodology [**This Volume**]
[SJ-TR-ARCH-9603A]

Volume VI — Additional Reports and Papers
[SJ-TR-ARCH-9603B]

* * *

CHAPTER I

INTRODUCTION

1.1 Background

The apparel industry in the United States has seen some drastic changes in the past decade and a half. Among the most significant are the massive influx of imports and the changes in apparel retailing. The domestic apparel manufacturing industry, which has been beset with problems of high cost and slow productivity growth, has suffered severely as a result of these changes.

The United States has gone from being a net exporter of garments in 1972 to a position in 1987 where imports accounted for more than 50% of the garment sales [Solinger80, AIT88]. From 1984 to 1987, the U.S. apparel manufacturing industry's share of the domestic market declined from 60% to less than 50% [AIT88]. Employment in this industry has steadily fallen from 980,000 in 1973 to about 650,000 today, mainly because of the loss of market share [Jayaraman88a]. The domestic apparel production of $39 billion in 1987 was 2% lower than the corresponding figure for 1984 after adjusting for inflation [AIT88].

The survival of the apparel manufacturing industry is of strategic importance to the nation: for the military during times of mobilization and on a broader scale for the nation's economy. The apparel and textile manufacturing sector, with an employment of more than 2 million, is the largest provider of manufacturing jobs in the country [Solinger80]. To regain its competitive position, the apparel industry needs to take a serious look at the way it conducts its business, and be prepared to make the changes that may be needed.

The apparel market has evolved into a highly fashion-oriented arena. The market

for apparel products has become more diversified because consumers lead more diversified and active life-styles, and seek specific products for specific end uses. The retailers have responded to this by analyzing their markets rigorously and targeting specific consumers with specific products. Consequently, order sizes are small and retailers are making their buying decisions as late in the season as possible. The turnover of styles is also much larger and fashion cycles are accelerating. The key to competing in such an environment is to achieve freedom from economies of scale and be in a position to manufacture goods in small batch sizes profitably and at short notice. This is particularly true for the differentiated merchandise end of the market which is dominating the other segments. In the fast changing market, the domestic manufacturers have an inherent advantage of being geographically close to the market. However, the industry is geared towards mass production and is unable to use this geographical proximity to the market to its advantage [TAC88].

Tackling the flexibility and response time problem are seen as the prime objectives of the industry's strategy to regain its competitiveness. This is a break from the traditional approach of high volume production to achieve lower cost through economies of scale. The objectives of this new strategy can be achieved through application of flexible automation concepts to apparel manufacturing. The term often used in the context of flexible automation is Computer-Integrated Manufacturing (CIM). CIM involves the use of automation techniques for the engineering, planning, manufacturing, and business functions of an organization, and integration of all these functions into a cohesive enterprise system through a shared information base. Thus, the scope of CIM is not limited to the manufacturing function; it encompasses the entire enterprise.

Effective application of CIM concepts to an apparel manufacturing enterprise requires a thorough understanding and analysis of the functions that the enterprise performs and the information that is shared among these functions [Jayaraman90]. Models provide the means for understanding the functions of the existing enterprise and for effectively

communicating the specifications of the proposed enterprise CIM system to those responsible for its design and implementation. A set of models that represents the relevant aspects of an enterprise's operations is referred to as the *architecture* of the enterprise. A comprehensive architecture of an enterprise may consist of models that capture its function and information structures, and its dynamic behavior.

The models representing the system as it currently exists make up the *AS IS* architecture whereas the models representing the proposed system constitute the *TO BE* architecture. Both, the AS IS and the TO BE architectures play important roles in the CIM system development cycle depicted in Figure 1.1. The AS IS architecture provides the means for analyzing the existing system. The TO BE architecture provides the specifications based on which the detailed design and implementation of the proposed system can be carried out.

1.2 Research Objectives

The objective of this research is to develop an architecture for an apparel manufacturing enterprise that will serve as a blueprint for applying CIM concepts to the enterprise. Although computer-based tools have been developed and used in the apparel industry to enhance the productivity of individual functional components of the apparel enterprise, these efforts have not been part of an overall scheme of integration. As a result, most enterprises operate with incompatible subsystems which cannot share information with each other. The current work is a maiden attempt at creating a systematic integration architecture for apparel manufacturing.

The functions and structure of an existing representative apparel enterprise have been studied and represented using descriptive modeling techniques. The resulting AS IS model has been analyzed to understand the integration needs of an apparel manufacturing enterprise. The results of the analysis of the AS IS model have been used to create the TO

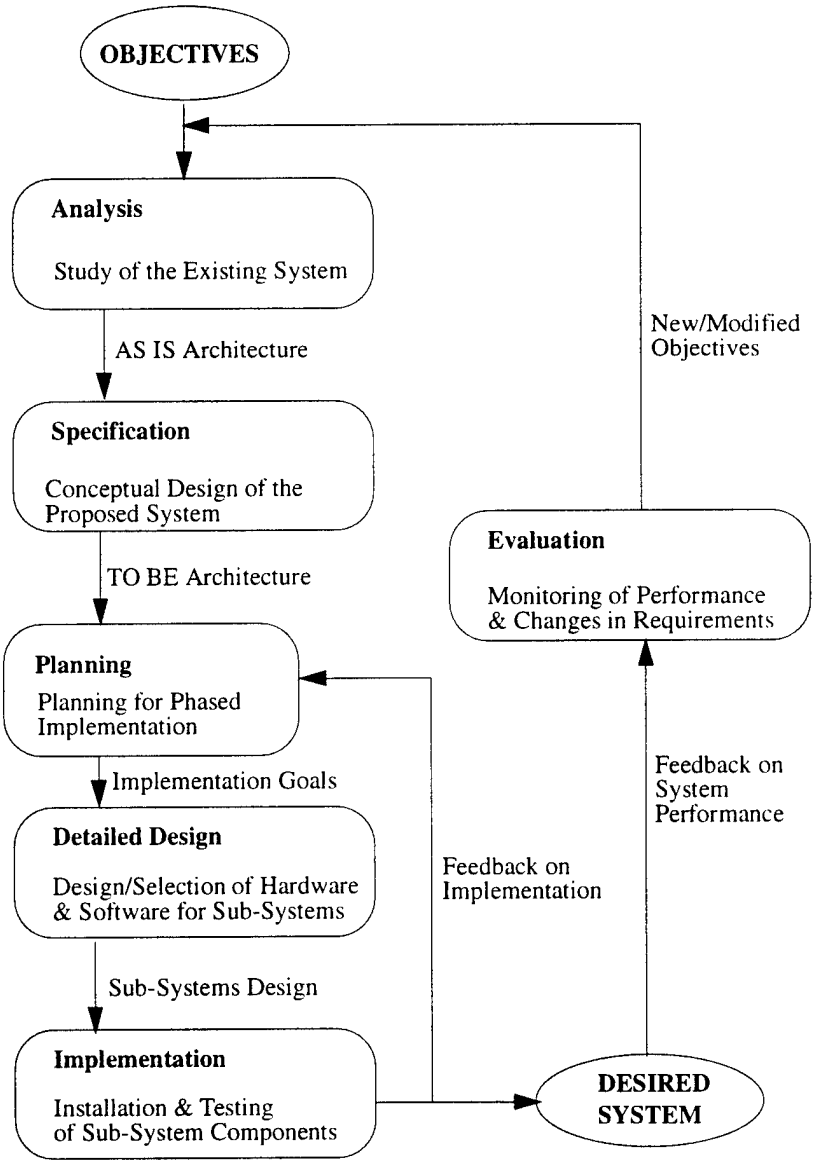


Figure 1.1. CIM System Development Cycle

BE architecture for an integrated apparel enterprise.

As part of this endeavor, modeling methodologies used for creating the architecture have been evaluated to determine their suitability for modeling manufacturing enterprises. An integrated framework for enterprise modeling (IFEM) has been proposed to overcome the shortcomings of the available modeling methodologies.

1.3 Overview

A critical review and evaluation of the available system modeling methodologies is presented in Chapter II. Chapter III outlines the procedure adopted to develop the apparel manufacturing architecture. The AS IS architecture is discussed in Chapter IV. The concepts of the proposed IFEM methodology for modeling manufacturing systems are outlined in Chapter V. Chapter VI, VII and VIII cover the TO BE information, function and dynamics models respectively along with the syntax of the various IFEM components used to develop these models.

CHAPTER II

SELECTION OF MODELING METHODOLOGY

A suitable modeling methodology is needed to document the operations of the existing enterprise in a precise and easily understandable manner and to develop an architecture for the proposed system. A good system design cannot be achieved without a sound modeling methodology. The design and analysis of any existing or proposed manufacturing system is often limited by the capability of the modeling methodology being utilized [Mackulak84]. If the manufacturing system analysis is not communicated to the management and production personnel effectively, it is doubtful that the suggested modifications or enhancements will be correctly incorporated into the system.

A methodology that has to provide the means for representation and analysis of manufacturing systems should meet the following criteria [Jayaraman90]:

1. It should be able to depict the operations of an enterprise in a natural and straightforward way.
2. Since the systems in question are vast and complex, the methodology should be structured and permit a hierarchical structure that lets models to be constructed and viewed at the desired level of detail.
3. Should be oriented towards graphical representation.
4. Should have a well-defined syntax for precise and unambiguous representation of the functions.
5. Should allow a wide range of users to communicate.
6. Should permit easy maintenance of the models.
7. It should be easy to design a software tool for modeling based on the methodology.

8. If there are multiple models, that complement each other, the methodology should provide means for maintaining consistency across the models.

In addition to the above mentioned criteria, availability of the methodology in public domain, support of the research community, and the cost and availability of the associated software tools are factors to be considered in the selection process. These criteria were used to evaluate the modeling methodologies reviewed in this chapter to ultimately select a suitable one for developing the AS IS and TO BE models.

2.1 Review and Evaluation of System Modeling Methodologies

A comprehensive specification of a modeling methodology has been provided by the Integrated Computer Aided Manufacturing program (ICAM) which was initiated in the 1970s by the US Air Force to improve productivity in the aircraft industry by systematic application of computer technology to manufacturing [ICAM81a]. ICAM recognized the development of a system modeling methodology as the first step towards achieving the project's goals. The ICAM project report notes that to get a comprehensive representation of the manufacturing enterprise being modeled, the methodology should be capable of producing the following:

1. A structured representation of the functions that the enterprise performs and the interconnections that exist between these functions.
2. A model of the structure of information needed to support these functions.
3. A model of the dynamic behavior of the components of the enterprise.

The models capturing the *function* structure, *information* structure and *dynamics* of a system are, together or individually, referred to as the *architecture* because they are used to understand, analyze and communicate how the various constituents of a manufacturing system fit together and interact.

A set of graphics-based methodologies developed by ICAM to model manufac-

turing systems was called IDEF (for ICAM DEFinitions). IDEF consists of three methodologies. Apart from the IDEF suit, there are other methodologies that have been developed for function, information and dynamics modeling. The IDEF and other available methodologies have been reviewed here to determine their suitability for developing an architecture for an apparel manufacturing enterprise.

2.1.1 Methodologies for Function Modeling

The IDEF methodology for function modeling is called the IDEF0 methodology and is presented in detail in the ICAM Function Modeling Manual [ICAM81a]. The other function modeling methodologies reviewed are the process flowcharting technique used by the European Strategic Planning for Research in Information Technology (ESPRIT) program [Yeoman85] and the data flow diagramming technique (DFD) [DeMarco78], which has been used extensively for functional analysis in information systems design.

$IDEF_0$ Function Modeling Methodology: $IDEF_0$ expresses the functions of a system and their interactions using a graphical language called the *cell modeling technique* [ICAM81a]. Each function is represented by a box and the *inputs*, *controls*, *outputs* and *mechanisms* associated with the function are drawn as arrows (Figure 2.1). The position at which the arrow enters a box conveys the specific role of the interface. The cell modeling technique is a hierarchical approach to modeling. An activity box, with its associated arrows, provides a bounded context for the activity that the box represents. The details of this activity can be represented as a diagram consisting of three to six boxes (each representing a sub-function of the activity being detailed) and interface arrows showing the interactions between these boxes. The diagram is restricted to the context provided by its parent box. Nothing can be added to or removed from this context while detailing an activity.

Such detailing of activities can go on till the desired level of detail is obtained. The function model consists of this set of inter-related diagrams, the accompanying textual

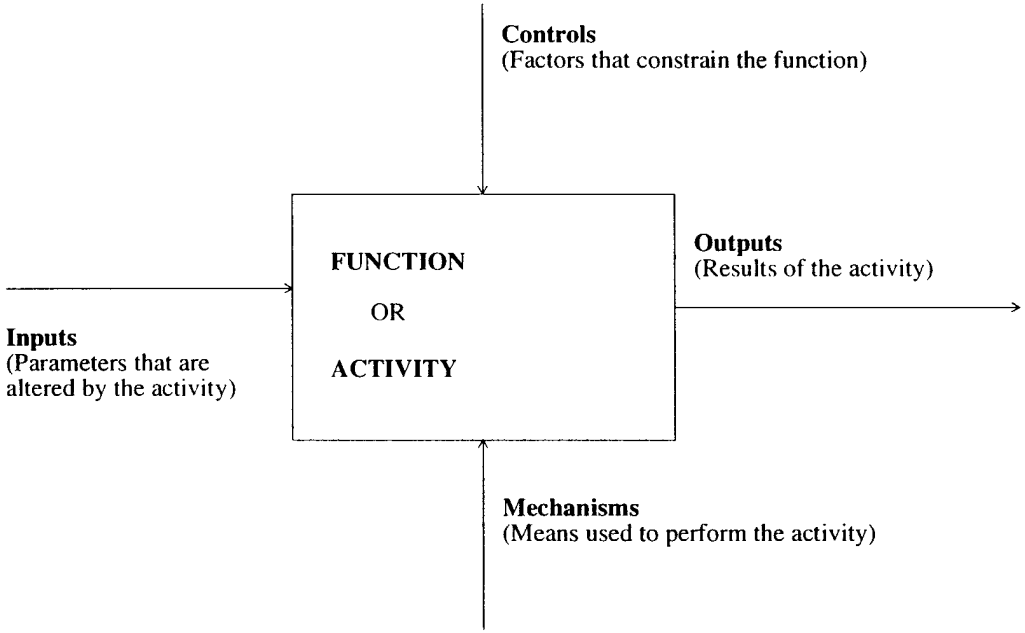


Figure 2.1. $IDEF_0$ Function Representation

description and the glossary of terms. At the top of the model hierarchy, is a diagram consisting of a single box and interface arrows representing the entire scope of the model. This diagram is called the context diagram. Also associated with a function model is the viewpoint that determines what can be "seen" within the context and from what "slant", and the purpose that establishes the intent of the model, i.e., the goal of communication it serves. The diagrams in a model are numbered according to their position in the hierarchy. The numbers begin with the letter A (for Activity or function) and the context diagram has the number A-0 (A minus zero). The context diagram is detailed into the A0 diagram and the diagrams representing the details of the boxes in the A0 diagram are numbered consecutively beginning with 1, i.e., A1, A2, etc. When the A1 box (function) is detailed, the resulting diagrams are numbered A11, A12, A13, and so on. To ensure usefulness and clarity, the methodology suggests that if a function cannot be subdivided into three lower level component functions, then the division should not occur. The IDEF0 models do not indicate precedence sequences or flows. The arrows do not signify any passage of time.

The methodology also specifies the procedures for collecting information for modeling and for critiquing the models before they are finalized. The information for modeling is collected by interviewing people closely associated with the activities being modeled. The review of the model is a cyclical process. The model is distributed to a group of experts in the area and they provide their feedback to the author of the model. The feedback is analyzed and incorporated into the model, which is sent for review again. The cycle is repeated till the model attains acceptable quality.

Process Flowcharts: The method used by the European Strategic Planning for Research in Information Technology (ESPRIT) program uses a flowcharting technique for documenting the functional structure of manufacturing enterprises [Yeomans85]. The flowcharts consist of a set of nodes connected by arrows depicting the flow of materials or information between the nodes (Figure 2.2). The nodes are one of the following type and

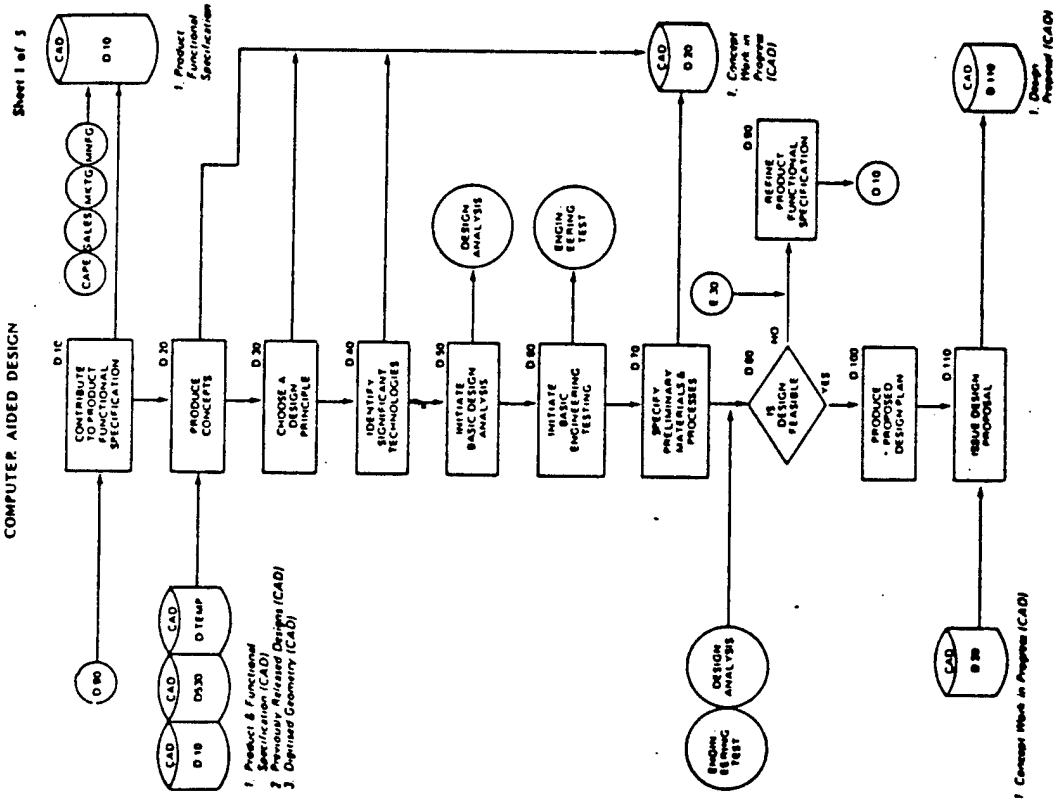


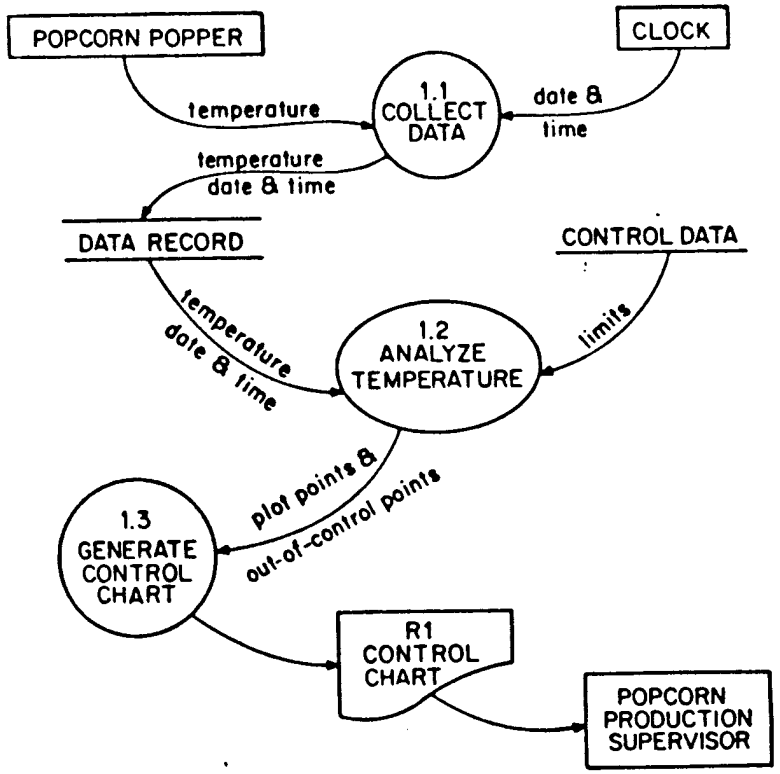
Figure 2.2. The ESPRIT Flowcharting Technique

are represented by different symbols:

1. A *process* (activity) is represented by a rectangle on a flowchart.
2. A diamond symbol is used to show a *decision* point in the processing flow. The decision point poses a question that can be answered positively or negatively, giving two alternatives for flow routes.
3. A hexagon symbol is used to represent an *initiating process* on the flowchart, e.g. a manual start-up.
4. A *connecting node*, represented by a small circle, shows the flow from a node on one flowchart to a node on another.
5. A cylinder symbol is used to identify *information* that is used by, or created by a *process* node.
6. Large circles represent a *parallel flow* of information in another flowchart. The name of the parallel activity is mentioned in the circle.

All the nodes in a flowchart are numbered using a letter specifying the area to which that activity belongs, and a sequence number. The major shortcoming of the flowchart technique is that it is not hierarchical, and does not allow perusal of the model at the desired level of detail (or abstraction). Also, the parallel flows are difficult to follow and the flowchart does not explicitly show what type of materials or information is flowing between the nodes.

Data Flow Diagramming: Another popular function modeling methodology that has been used in system modeling and analysis is the Data Flow Diagramming (DFD) technique [DeMarco78, DeMarco79, Gaylord87, STRADIS88, Perkinson84]. DFD methodology has been used extensively in large information system design projects to carry out functional analysis that precedes data analysis and modeling. DFD graphically describes the functional structure of the system as a network of activity nodes and data flow vectors (Figure 2.3). Additional objects like information sources and sinks, files and documents are also



Detailed popper control system data flow.

Figure 2.3. The Data Flow Diagramming (DFD) Technique

used as termination points for the data flows. There is a different symbol associated with each object, and the objects are labeled with unique names. The model constructed using DFD is hierarchical in nature because each activity node can be expanded into a network providing details of that activity. DFDs are supported by "minispecs" and data dictionaries. Minispecs provide a complete but concise description of each activity in a page or less [Piltzecker85]. Various formats, including structured English, truth tables, fault trees, or ladder diagrams are acceptable for writing minispecs. Data dictionaries provide precise qualitative and quantitative definition of each data label used in the DFDs. Data dictionaries enforce universal definitions of data for system-wide use. Unlike $IDEF_0$, DFD does not explicitly distinguish between inputs and controls.

2.1.2 Information Modeling Methodologies

The information model defines the structure of the data that is maintained by the enterprise being modeled to support its functions. Information is data to which meaningful structure has been imparted. The information maintained by the enterprise is an abstract representation of the real-world entities that the enterprise deals with, e.g., purchase orders, designs, schedules, etc. An effective methodology for information modeling must provide the means for capturing the real-world semantics precisely in a form that is easy to understand. The methodology should also permit creation of a single integrated definition of the data, within the enterprise being modeled, which is unbiased toward any single application of data and is independent of how the data is physically stored or accessed. This view of the data, called the "conceptual schema", helps maintain the consistency of data across the applications.

The conceptual schema extends the two-schema approach that has traditionally been used for data modeling for DBMS applications. In the two-schema approach the data is defined from two distinct views, the user view and the computer view. The user view,

referred to as the "external schema", defines the data as seen by an individual application that uses the data. Since the same data objects may be used differently by different applications, the user view of the data is often inconsistent across the enterprise. The computer view of the data, or the "internal schema", defines the data structures for storage and retrieval of the data. The internal schema is hardware dependent. The conceptual schema approach is required when the data is defined for the whole enterprise, and not for specific applications inside it.

On the other hand, the information model must also conform to the database model used to implement the information system which maintains the enterprise data. In fact, the available information methodologies have been developed for a particular database implementation paradigm. The relational model of data attributed to Codd [Codd70] is the basis of many available information modeling methodologies, including the IDEF information methodology, $IDEF_1x$. DBMSs based on the relational model are widely used in commercial applications. A recent development in database technology is the advent of semantic or object-oriented DBMSs that abstract the real-world entities as data the way humans do. Information modeling methodologies based on both, the relational and object-oriented paradigm are reviewed.

$IDEF_1x$ Information Modeling Methodology: $IDEF_1x$ [IISS85] is a methodology for representing the structure of the data needed to support the functions of the manufacturing enterprise. The methodology is based on the relational paradigm and is derived from the Entity-Relationship Approach developed by Chen [Chen76]. In an $IDEF_1x$ model, the *entities* about which data is maintained are defined in terms of their characteristics. The characteristics are represented as *attributes* in the model, and entities as aggregates of attributes. For each entity, the set of attributes that uniquely identify an instance of that entity are represented as the primary key of the entity. The relationships between entities are expressed by migrating the *primary key* attributes of one entity to another. The attributes in-

herited by an entity from a related entity make up the *foreign key* of the entity. In the model, the entities are graphically represented as boxes and relationships as directed lines joining the boxes. The attributes of an entity are listed in the box representing the entity. The model is normalized to the third normal relational form [Codd70].

The limitations of the relational model in capturing the semantics of the real-world entities have been discussed by Bowers [Bowers89]. Because of the limitations of the underlying relational model, the IDEF$_1$x methodology suffers from a few major shortcomings. Firstly, the rules of normalization do not permit the attributes to have multiple values which is often the case in real-world entities. For example, one of the attributes of a *purchase order* is the list of items ordered. Since multi-valued attributes are not permitted, a purchase order has to be modeled by creating a relationship entity *purchase order item* each instance of which represents a purchased item. This results in fragmentation of a real-world entity into multiple IDEF$_1$x entities. The clarity of the model is reduced when real world-entities have to be expressed using complex joins of IDEF1x entities using some form of predicate calculus. Secondly, the methodology does not provide any means of expressing the constraints that reflect the semantics of situation being modeled. For example, a constraint such as *the salary of an employee cannot exceed that of his supervisor* cannot be modeled using IDEF$_1$x.

Another major weakness of the model is the use of foreign keys to express relationships. In the IDEF$_1$x models, the primary key for each entity has to be identified so that relationship of the entity with other entities may be expressed by migrating the primary key to the related entities. Using primary key values to identify an instance of an entity implies that an entity instance exists only if the primary key attribute values for it are known. This implication may run contrary to the semantics of the situation being modeled. For example, a purchase order may come into existence and be fully prepared before a PO number, its primary key attribute, is assigned to it. This situation cannot be modeled using IDEF$_1$x

since the primary key concept implies that a purchase order can exist only if a PO number for it exists.

Information Models Based on Extended Relational Models: Extensions have been proposed to the original relational model to overcome some of its shortcomings. Most of these extensions remove the restriction imposed by the first normal form [Codd70] that attributes be single valued. These models are referred to as the NF^2 data models since they drop the first normal form restriction [Dadam89]. One such model is the extended relational model RM/T proposed by Codd [Codd79]. In this model, an entity is represented as an entity relation (E-relation) which contains a single column *surrogate* value that identifies an instance of the entity uniquely. Attributes are represented by a property relation (P-relation), which associates surrogate values with property values. A P-relation can be assigned a set of surrogates, thus permitting an attribute to have a set of instances of another entity as its value. In the purchase order example, the item attribute of the entity purchase order can have a set of instances of entity purchase order item as its value.

By permitting attributes to have relations (tables) as values, the RM/T model permits hierarchical structures, in which entities can be defined as aggregates of other entities. While most of the shortcomings of the relational model are overcome, the constraints on the attribute values still cannot be expressed using the extended relational models.

Object-Oriented Information Models: Object-Oriented models of data aim at producing a user-understandable definition of data and capture a substantial part of the semantics of the real-world situation being modeled [Garvey89]. In object-oriented models, data is structured into object classes that conform to the real-world entities being modeled. The attributes of the object classes are viewed as functions, instead of just data items, thus allowing constraints to be captured in the definition of the attributes. Structured data models can be created by defining classes as aggregates of other classes. Creation of an aggregation hierarchy is made possible by permitting attributes to have sets of other objects as

their values, as in the extended relational models. A distinguishing feature of the object-oriented data models is inheritance, whereby classes can be specialized into sub-classes that inherit the attributes of the super-class from which they are specialized. For example employees can be specialized into managers, supervisors and workers. The attributes common to all employees are contained in the employee class and inherited by its specialization classes manager, supervisor and worker. These three classes only contain attributes that are unique to them.

The semantic data model (SDM) of Hammer and McLeod [Hammer81] is an example of an object-oriented information model and is implemented by many recently introduced or still experimental object-oriented DBMSs (OODBMS), such as the InfoExec system from Unisys [Balfour90]. Other object-oriented data models available are conceptually similar to SDM. Among these are the IRIS model [Lyngbaek86] which has been implemented in the IRIS OODBMS [Wilkinson90], the DAPLEX model [Shipman81] and the POSTGRES model [Rowe87, Stonebraker87]. A graphical modeling technique for creating object-oriented data models is illustrated by Schrefl [Schrefl89]. Although the object-oriented models capture the semantics of the modeled domain very well, the DBMSs implementing this paradigm are still in the experimental stage. It is not clear whether information models created using the object-oriented paradigm can be implemented effectively.

2.1.3 Dynamics Modeling Methodologies

The dynamics modeling methodology developed for the ICAM project is known as $IDEF_2$ and is discussed in detail in the ICAM Dynamics Modeling Manual [ICAM81b]. The other choices in dynamics modeling methodologies come from the field of discrete-event simulation. A variety of commercial simulation software is available and each package provides a methodology of some sort for modeling the system whose behavior is to be analyzed through simulation.

$IDEF_2$ Dynamics Modeling Methodology: IDEF2 [ICAM81b] provides a vehicle for describing the elements of manufacturing system whose behavior varies over time. The fundamentals of $IDEF_2$ are based on computer simulation modeling techniques. To describe a system in $IDEF_2$, the system is modeled as four submodels:

1. The *facility submodel* describes the resources which are used by the system to produce its outputs. These resources are classified as physical, logical and cognitive. Materials, people and machines are the physical resources, whereas the procedures and software that guide them are the logical resources. Cognitive resources are those resources which are required for thought processes such as experience and creativity.
2. The *entity flow submodel* graphically describes the flow of products and information through the facilities described by the facility submodel. This submodel describes the activities performed to obtain the flow and the decisions regarding alternate flow of entities.
3. The *resource disposition submodel* is used to describe the disposition of resources when they become available. A resource in $IDEF_2$ is any part of the system which must be present to perform an activity. The model associates different actions to the different dispositions of the resources.
4. The *system control submodel* describes the occurrences of activities which control but do not prescribe the flow of entities. Situations handled by this submodel include the breakdown and repair of resources, the arrival of entities, the alteration of resource capacities, the initiation and termination of shifts and the alteration of job priorities.

The heart of the $IDEF_2$ model is the entity flow sub-model that represents the dynamic actions that produce the flow of an entity through the system being modeled. Although the methodology provides a thoroughly detailed description for simulation, the

model generated is very large in size even for a small system because of lack of facilities for abstraction and parameterization [Godwin89]. This is analogous to having a programming language without subroutines and parameters. With the lack of parameterization and abstraction, the process sequences have to be explicitly built into the entity flow networks. As a result, complex entity flow routes in which many alternative paths exist at each node are difficult to model. Also, it is virtually impossible to model flexible manufacturing systems in which a variety of entities, each with a different process sequence, is concurrently processed by the system. Thus, the $IDEF_2$ methodology does not meet the requirements for modeling the dynamics of CIM systems.

Dynamics Modeling Alternatives: Since dynamic models provide the system description for computer simulation, dynamic modeling methodologies are closely associated with the computer languages and tools available for system simulation. A dynamics model of the system has to be built before it can be simulated on the computer. GPSS [Gordon78, Schriber88] is the oldest general purpose simulation language around. In GPSS and its various implementations, the system is modeled as a waiting line with servers. A block diagram is used to represent the system as a model. For simulation, the information on the block diagram needs to be manually converted into GPSS code.

The other popular general purpose simulation languages are SIMSCRIPT [Russell88], SLAM [O'Reilly88] and SIMAN [Pegden82, Suri88]. Each language has its own methodology for modeling the system to be simulated. Among these languages, SIMAN is claimed to be designed specifically for simulating manufacturing systems [Davis88]. It provides features that facilitate the modeling of typical manufacturing sub-systems, such as material handling equipment. SIMAN uses a graphical representation for the system model. Another distinguishing feature of SIMAN is its separation of the system model from the data required to run a particular simulation experiment [Banks88]. This feature helps prevent accidental changes to the model when changes are made to the simulation conditions

for different simulation experiments. An accompanying package called CINEMA [Miles88] provides animation capabilities for SIMAN.

Some of the limitations of general purpose simulation languages in modeling manufacturing systems are discussed in the literature. Young et al. observe that the available general purpose simulation languages are designed to support only the modeling and analysis of process and material handling flow, and do not cater to all the requirements of CIM system modeling [Young88]. Adiga and Glassey point out the limitations of such languages in modeling complicated decision-making algorithms and logic related to selection of entities from queues [Adiga88]. Another limitation of these languages is the lack of facilities for integration into a database environment [Pruett90]. Consequently, the dynamics models created using general purpose simulation languages cannot use the entity structure definitions contained in the associated information model.

2.2 Selection of Modeling Methodology

The selection criteria discussed earlier in this chapter were used to evaluate and select suitable methodologies for developing the AS IS and TO BE architectures. The selection of the methodologies for the AS IS and TO BE architectures is discussed separately as the factors influencing the selection in the two cases were different.

2.2.1 Methodology for AS IS Architecture

$IDEF_0$ function modeling methodology was selected for modeling an existing apparel enterprise to gain an understanding of the apparel manufacturing domain. The $IDEF_0$ methodology was selected over the flowcharting technique of ESPRIT because of its hierarchical structure that permits top down model development and a gradual exposition of the details of the model. $IDEF_0$ also has some advantages over DFD, which was the other hierarchical modeling technique considered. $IDEF_0$ has a recommended set of methods for

data gathering, modeling, model review and maintenance which DFD lacks. $IDEF_0$ also provides explicit means of expressing the mechanisms for carrying out the activities. Therefore, IDEF0 was preferred to DFD as the methodology for function modeling.

The AS IS architecture consists only of the $IDEF_0$ function model as this model contains sufficient information to understand how an existing apparel enterprise functions. The model provides a detailed breakdown of the functions of apparel manufacturing. It also depicts the interconnections that exist between these functions. The model glossary provides the meaning of the information that is shared between the functions through the modeled interconnections.

2.2.2 Methodology for TO BE Architecture

The TO BE function, information and dynamics models make up the architecture of a CIM system for an apparel enterprise. Since the architecture consists of multiple models that complement each other, the selection of modeling methodologies was strongly influenced by the model integration criterion.

$IDEF_0$ was selected for the TO BE function model for reasons discussed in the previous section. In addition, $IDEF_0$ was also found suitable to serve as the foundation for a framework into which the three models could be integrated. Model integration could be facilitated by using the $IDEF_0$ function model as a context for the dynamics model and correlating the entities defined in the information model with the arrows that represent interfaces between the $IDEF_0$ functions.

Since, the information model serves as a conceptual design for a database system, the methodologies for information modeling are closely related to the database model in which the system is implemented. Whereas the methodologies based on the object-oriented paradigm provide models that are closer abstractions of the real-world than models based on the relational paradigm, the object-oriented database technology is still in an experimen-

tal stage. On the other hand, the relational database technology is mature and well-established, albeit lacking means to capture the rules that constrain the data. Therefore, the choice for an information methodology was not very clear-cut. $IDEF_1x$, methodology, which is based on the relational model, was selected for information modeling since the $IDEF_1x$ models can be used readily to define the relational database schema for a CIM system. The IDEF1x model was extended to integrate the information model with the function model. These extensions form a part of the integrated framework for enterprise modeling (IFEM) that has been developed as part of this research and used for the TO BE architecture. IFEM is discussed in detail in Chapter V.

In the TO BE architecture, the dynamics model provides the means for analyzing the dynamic behavior of the integrated apparel manufacturing system whose static design is represented by the TO BE function and information models. The $IDEF_2$ and other simulation methodologies reviewed in this chapter were found unsuitable for developing the TO BE dynamics model for various reasons. First, the models produced by these methodologies are stand-alone models that duplicate the contents of the function and information model. As a result, it is not possible to integrate these models with the function and information models. Also, these methodologies were designed to model the flow of a particular entity through the system. Hence, these methodologies cannot be easily used to develop generic dynamics models of the enterprise, that are independent of the flow of a particular entity through the enterprise. For developing the TO BE dynamics model, the $IDEF_0$ function modeling methodology has been extended to represent the dynamic behavior of the functional components of the enterprise. These extensions form part of the proposed IFEM methodology discussed in Chapter V.

CHAPTER III

RESEARCH PROCEDURE

The development of the architecture for CIM system for apparel manufacturing was carried out in three phases. In the first phase, the scope of the architecture was defined. The AS IS model of an existing enterprise within the defined scope was developed next. In the third phase, the AS IS model was used as the basis for developing the TO BE architecture which represents a completely integrated apparel enterprise.

3.1 Scope of the Model

The scope of the architecture is stated in terms of the objectives of the modeling activity (purpose), the boundaries of the domain under consideration (context) and the perspective from which the domain is seen for modeling purposes (viewpoint). The scope needs to be stated clearly to avoid cluttering the model with superfluous information that obscures its relevant contents. General information about the apparel manufacturing domain was gathered through literature, plant visits and interviews with people involved in apparel manufacturing. Based on the gathered information and the research objectives, the scope of the architecture was defined.

3.1.1 Purpose

The purpose of the architecture of an apparel manufacturing enterprise is to provide an understanding of the range of activities involved in the day-to-day operations of an enterprise and serve as a blueprint for implementing CIM in the enterprise. The AS IS model serves as the means for understanding how an existing enterprise functions whereas the TO BE models make up the architecture of the proposed CIM system for apparel manufac-

turing.

3.1.2 Context

A primary consideration in the development of the architecture is that the architecture be representative of the general operational characteristics of the apparel industry. For this purpose, the approach adopted for AS IS modeling was to thoroughly study and model one representative enterprise, have the model reviewed by a broad cross-section of the industry and refine it for correctness and completeness. Trouser manufacturing was chosen as the initial target domain because trouser manufacturing represents generic apparel manufacturing domain well and real-world data for this domain was made accessible by an industry partner. The TO BE architecture extends the AS IS model to cover generic apparel manufacturing. Activities ranging from marketing and product development to finished goods distribution are included in the context.

3.1.3 Viewpoint

Based on a preliminary analysis of the apparel industry, the activities performed by an apparel manufacturing enterprise were divided into three main categories:

1. Strategic decision-making related to the long-term strategy of the enterprise. This function is performed by corporate management. Investment and expansion decisions are examples of long-term decision-making.
2. Tactical decision-making related to the day-to-day workings of the enterprise. These decisions are made by the middle level management (e.g., departmental and plant managers). Examples of these decisions are production planning, inventory management, manufacturing, quality control, etc.
3. The third category of activities is the operational level activities whereby the tactical decisions are implemented. For example, the activities involved in assembling a garment fall into this category.

The model takes the viewpoint of the middle level managers responsible for day-to-day decisions in the enterprise. Strategic decision-making activities were excluded from the model since the outcome of these decisions are policies that usually do not change on a day-to-day basis. Thus the viewpoint of the model was restricted to tactical decision-making and operational functions of the enterprise.

3.2 AS IS Model Development

For developing the AS IS model, this research depended heavily on interaction with the apparel industry to obtain the modeling data from real-world apparel enterprises. Oxford Slacks of Monroe, Georgia, a member of the Apparel Manufacturing Technology Center (AMTC) Steering Committee, served as the primary source of data for the model. Other apparel industry sources participated and assisted in the model review process.

3.2.1 Information Gathering

The information for building the AS IS function model was gathered through regular meetings with the managers of the different departments at Oxford Slacks. Beginning with the overall view of the organization, the departments associated with the various functions of the enterprise were covered. Typically, at each meeting with the department manager, the activities performed by the department and the inter-department interaction associated with these activities were discussed. A part of the interview sessions were devoted to recording managers' comments on current practices and suggestions for improvements. The paperwork involved in the inter-departmental interactions was also gathered and studied to understand the information flow associated with each activity. After the interview session, a written summary of the gathered information was submitted to the manager for review. The manager's comments and correction were incorporated into the written summary.

3.2.2 Model Development and Review

Concurrently, the development of the AS IS function model was carried out. A glossary of data objects, represented by the interface arrows on the IDEF0 diagrams, was also developed to support the model. The completed AS IS model and its glossary were presented to Oxford Slacks for review and based on their comments, the model was refined. The model was also distributed to other industry members for review. The purpose of the external review was to ensure the completeness of the model and correctness and applicability of the terminology used in the model glossary.

3.2.3 Software Tools for Modeling

The AS IS function model was developed using the IDEFine-0 modeling software from Wizdom Systems Inc. [Wizdom88] IDEFine-0 provides tools for drafting IDEF0 node diagrams and checking their syntax. The glossary of the model was developed and maintained separately using a word processor since the glossary building tools of IDEFine were found to be inadequate.

3.3 TO BE Architecture Development

The TO BE architecture of the apparel enterprise was developed through analysis and refinement of the AS IS architecture. It consists of function, information and dynamics models of an apparel manufacturing enterprise, which were integrated into a single framework to serve as a comprehensive architecture for a CIM system for apparel manufacturing.

3.3.1 Information Model Development

The TO BE information model was developed to serve as the database schema for the apparel CIM system. The model provides a coherent definition of the data maintained by the enterprise, based on the semantics of this data. The information model, developed using the IDEF$_1$X methodology, consists of a set of entity-relationship diagrams and a glos-

sary. Entities about which data is maintained, their attributes and the relationships between these entities were identified from the glossary of the AS IS architecture and represented graphically on the diagrams. Definitions of the entities and their attributes were documented in the glossary.

The TO BE information model was developed using the IDEFine-1 software from Wizdom Systems Inc. [Wizdom88]. The glossary was developed using a word processor. Since IDEFine-1 did not provide any means for checking the integrity, a software utility for this purpose was developed. This utility was also enhanced to translate the model into data definitions for an SQL relational database such as ORACLE [Oracle88].

3.3.2 Function Model Development

The TO BE function model was developed to provide the functional structure for a CIM system for apparel manufacturing. The model represents the functions of an apparel manufacturing enterprise as distributed components of a CIM system. The data interfaces between the functions were modeled using the data definition from the TO BE information model.

The base TO BE function model was developed using the IDEF0 methodology. Extensions to the IDEF methodology, developed as part of the IFEM methodology, were used to integrate the TO BE function model with the information model. The model was developed using the AI0 modeling software from Knowledge Based Systems, Inc. [KBSI89]

3.3.3 Dynamics Model Development

The TO BE dynamics model was developed to represent the dynamic behavior of the functions modeled in the TO BE function model. The model was built on the foundation provided by the function and the information models using the proposed IFEM methodology. In the TO BE dynamics model, the conditions that activate each function and the dy-

namic interactions between the inputs, controls, outputs and mechanisms of the functions were modeled.

The AS IS and TO BE architectures and the specific details of methodologies used to develop these architectures are discussed in the following chapters. Although the three component models of the TO BE architecture were developed in the order in which they are presented, they were refined through numerous iterations.

CHAPTER IV

AS IS FUNCTION MODEL

The first step in developing an architecture for implementing CIM in an apparel enterprise is to gain a thorough understanding of the apparel manufacturing domain through a model of the functional structure of an existing apparel manufacturing enterprise. The functional structure of an existing apparel enterprise is captured in the AS IS function model discussed in this chapter. The glossary accompanying the model provides the description of the information processed by the enterprise functions.

4.1 An Overview of Apparel Manufacturing

Apparel manufacturing enterprises differ in size and complexity from small contract sewing shops to large corporations with their own design studios and product lines. There is a trend in the industry towards retailers' private brands for which the manufacturers act as suppliers. The enterprise modeled here carries out activities ranging from design to distribution of garments according to customers' specifications. At the top level, the functions of such an enterprise may be classified as follows:

1. Marketing, Product Development and Sales
2. Planning and Preparation for Production
3. Manufacturing
4. Customer Service
5. Distribution
6. Engineering and Quality Control Services

The *marketing, product development and sales* functions cover all the activities

performed before production orders are finalized. The enterprise works with the customer to develop garment styles according to customer's specifications. The sales people negotiate the contract with the customer for orders on garments that have been developed. This contract provides general terms of agreement between the enterprise and the customer and does not contain all the details such as color/size distribution of garments.

The central coordination point for all the activities after a sales contract has been signed, is the *customer service* function. It interacts with the customer on a regular basis and issues the production orders to the enterprise for meeting the customer's order requirements and schedule. It is the responsibility of customer service to ensure that the distribution system has the appropriate finished goods to ship to the customer when the customer sends the shipping orders.

The *manufacturing* function services the distribution function by supplying the finished goods. The *production planning and preparation* function, in turn, services the manufacturing function by scheduling production and making raw materials available. The *engineering and quality control services* functions are auxiliary functions providing services to other functions. The enterprise, as a whole, can be viewed as a provider of goods and services to its customers. The customers include chain stores, outlet retailers and mail order houses.

An important characteristic of a majority of the apparel manufacturing enterprises is that they do not produce to meet projected sales. All the production is for confirmed sales orders and the AS IS model reflects this characteristic. The enterprise should be prepared to produce goods at very short notice if it has to produce to firm orders. The customers desire the flexibility of making their buying decisions as late as possible. Even when sales contracts have been signed for the whole season, the enterprise does not know the specifics such as size and color distribution well into the season. The raw material (fabric, trim and accessories) suppliers play a critical role in the enterprise's ability to manufacture goods at

short notice.

The operations of a typical apparel enterprise are spread over a vast geographical area. The styling studios and marketing operations are located close to fashion centers, such as New York City, Dallas and Los Angeles. Manufacturing facilities are generally located in areas where inexpensive labor is available. Product development, pattern and marker making, and cutting are carried out in centralized locations which support numerous sewing and finishing plants, some of which are located overseas. Distribution of finished goods is carried out from a central location.

4.2 IDEF0 Diagramming Conventions

The AS IS model of a representative apparel manufacturing enterprise, as it appears within the context and viewpoint defined in chapter III, has been developed using the IDEF0 function modeling methodology. The model consists of a set of indexed diagrams and an accompanying glossary of terms. The diagrams model the enterprise in terms of the functions that the enterprise performs and their inter-connections. Functions are represented as boxes on the diagrams and described by short verb phrases. Each diagram contains from 3 to 6 boxes thus maintaining clarity and readability. All the interfaces to a function are represented by labeled arrows which connect to the rectangular function boxes from all four sides (Figure 2.1). The arrows from the left (*inputs*) represent inputs to a function and the arrows coming out from the right (*outputs*) represent the outputs that the function produces by transforming its inputs. The entities that constraint or control this transformation of inputs to outputs are represented as arrows coming in from the top (*controls*). The arrows coming in from the bottom (*mechanisms*) represent the mechanisms, i.e., the means used to perform the function. The interface arrows are also referred to as the ICOMs and are described in detail in the ICOM glossary. Outputs from one function can be inputs or controls for other functions. Arrows inter-connecting function boxes represent interfaces between

functions.

Where more detail on a particular function, represented as a box on a diagram, is desired, that box is decomposed on another diagram to depict its sub-functions, interfaces and inter-connections again as boxes and arrows. Thus the diagrams are tied in a hierarchy in which the higher level diagrams contain more abstract functions and the lower level ones go into greater details of each activity. Each diagram is bound by the context of its parent function box, i.e., all the arrows on the parent box connect to the arrows in the corresponding child diagram. The model hierarchy converges to a diagram at the top called the model context diagram which contains a single box and its interfaces. The interfaces represented in this diagram serve as the context for the whole model.

A node numbering scheme is used to number the diagrams and boxes. The context diagram is numbered A-0 (A minus zero) and is decomposed into the A0 diagram, which is the top level diagram depicting the major functions of the enterprise. Within a diagram, the boxes are numbered starting with number 1. The node number of a diagram is derived by appending the number of the box that the diagram details to the node number of the diagram in which this box belongs. For example, the diagram detailing box number 2 on diagram A32 is numbered A322. The partial node tree diagram of the AS IS function model shown in Figure 4.1 illustrates the node numbering scheme.

4.3 The AS IS Function Model

The AS IS model of the apparel enterprise consists of IDEF0 diagrams and a glossary [Malhotra90]. The A-0 diagram in Figure 4.2 shows the entire scope of the model, i.e., *Operate an Apparel Manufacturing Enterprise*. It establishes the boundaries (context, in IDEF terminology) of the model and forms the basis for further decomposition efforts. It also states the viewpoint from which the enterprise is viewed for modeling and the purpose that the model is intended to serve.

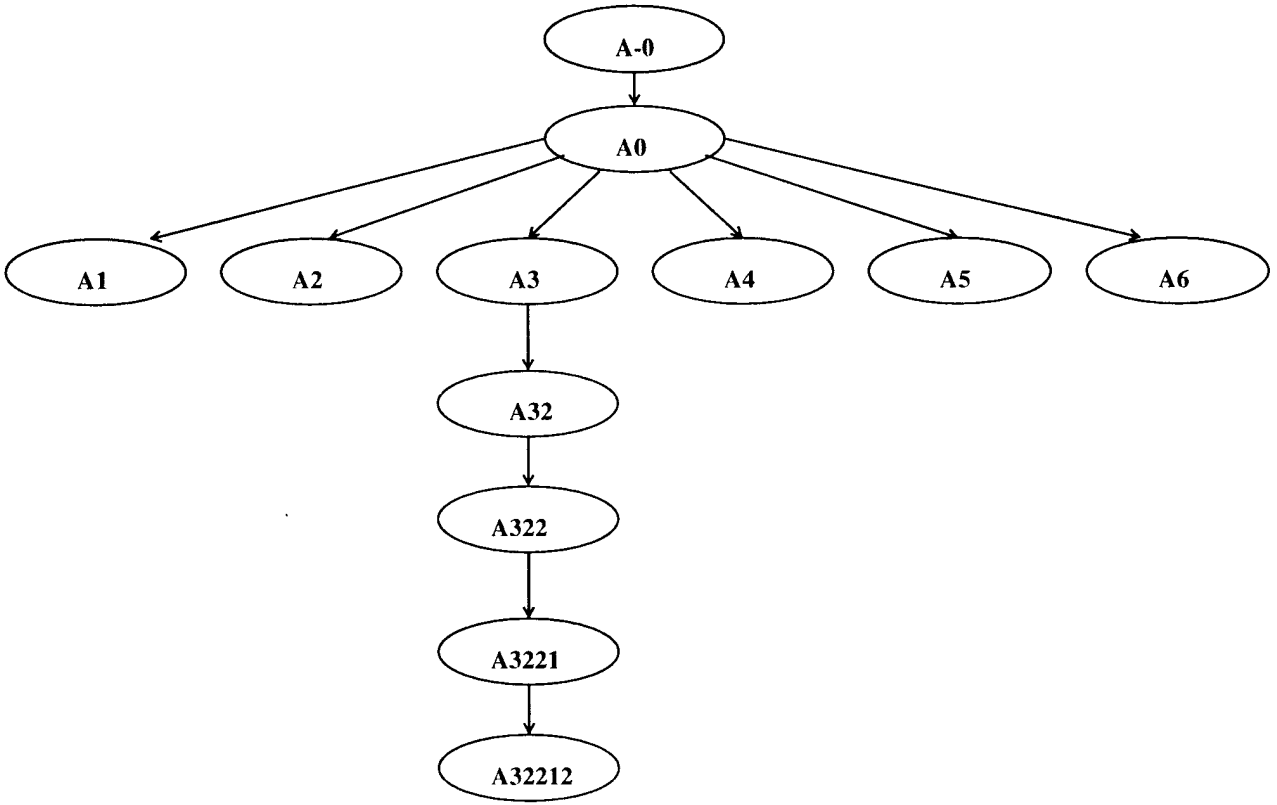


Figure 4.1. A Part of the AS IS Function Model as a Node Tree Diagram

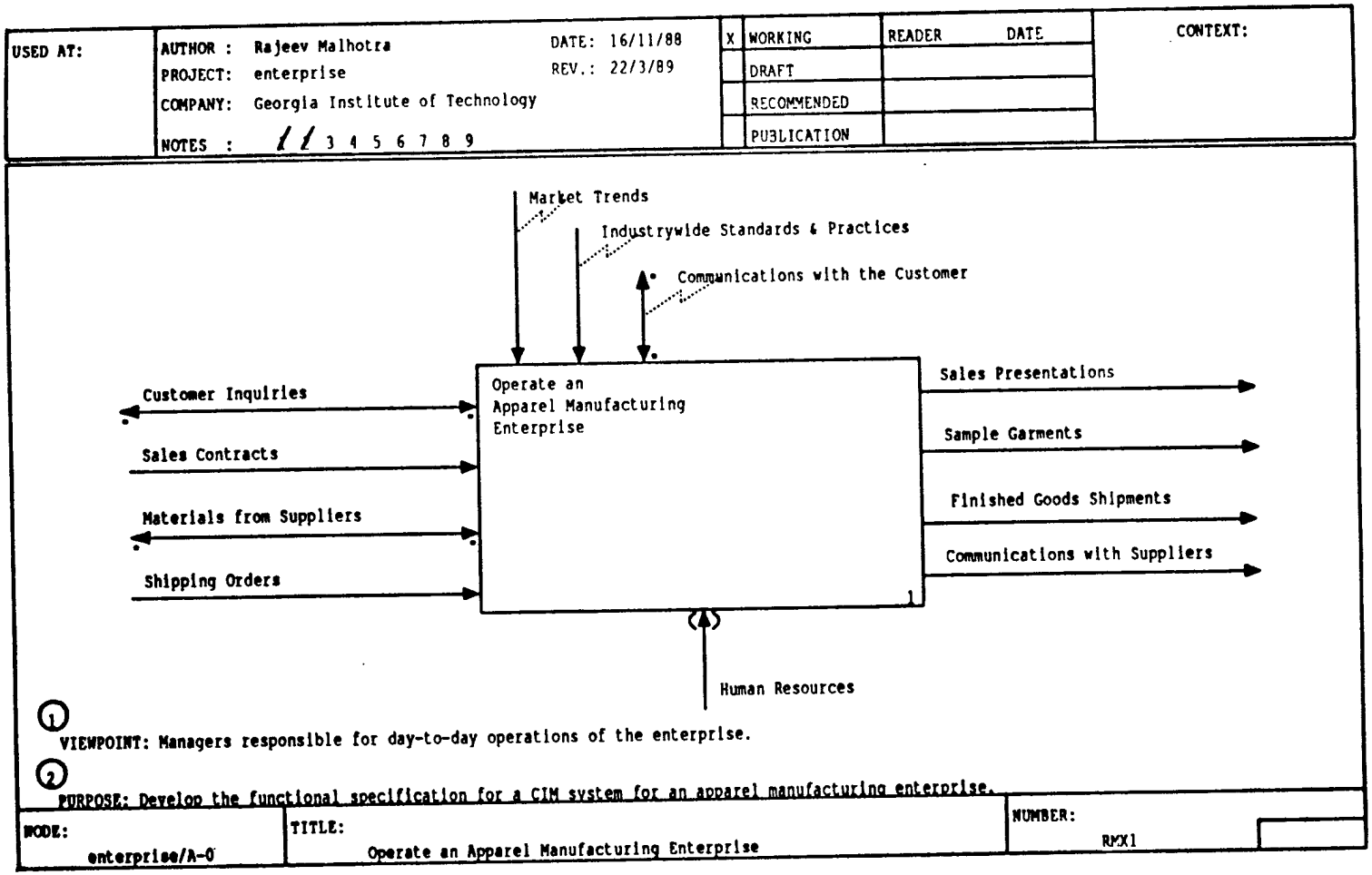


Figure 4.2. The A-0 Context Diagram

The inputs to the enterprise are denoted by the arrows to the left of the box. The enterprise receives inquiries from the customer on new products, designs and styles, and it in turn responds to the requests (hence the double-headed arrow with the dots). It receives sales contracts and shipping orders from the customer while materials (e.g., fabric and trim) are received from suppliers (materials may be returned if they do not meet quality standards). The operation of the enterprise is constrained or governed by market trends, industry standards and practices and the requirements of the customer. The resources or mechanisms responsible for the operation are humans and machines. The tunnels "()" around the mechanism arrow imply that the details of the mechanism are not important at the next lower level. The outputs from the enterprise are sales presentations, samples and shipment of finished goods to the customer, and communications with suppliers regarding materials.

4.3.1 The A0 Diagram

The decomposition of the A-0 diagram is shown in Figure 4.3. The A0 diagram clearly identifies the six major functions performed in the day-to-day operations of the enterprise. Since it provides a complete description of the model (including the interactions between the various functions), the A0 diagram is commonly referred to as the top level diagram. All the inputs, outputs, constraints and mechanisms on the A-0 diagram connect to their corresponding boundary arrows on the A0 diagram.

The first function (the first box) is to develop the garment for manufacturing and it is based on customer needs and market trends. There is a great deal of interaction between the enterprise and the customer during the course of product development and this is denoted by the double-headed (feedback) arrow. Materials are procured from suppliers to produce samples. The other results (outputs) of this activity are garment designs and sales presentations to customers.

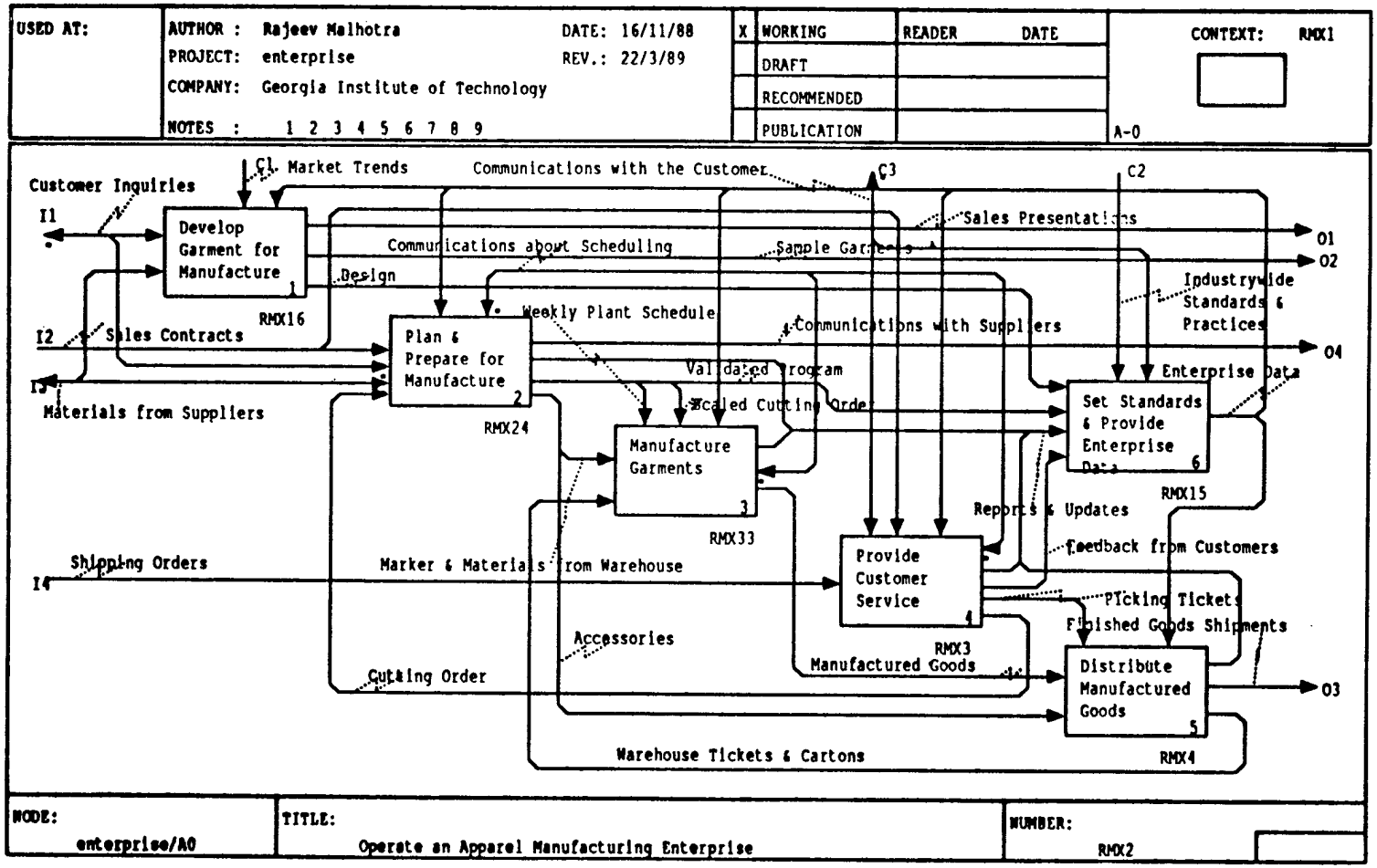


Figure 4.3. The Top Level Diagram of the AS IS Model

Once the customer places an order, the next function is to plan and prepare for manufacture and this is represented by the second box in the A0 diagram. This activity can be performed only if sales contracts have been finalized and the availability of materials ensured. It is constrained by the customer's delivery requirements. The outputs of this activity include the cutting schedule for the orders, communications with vendors regarding materials, and markers and materials from warehouse.

The third box represents the manufacturing function which encompasses cutting, sewing, finishing and garment inspection. It requires markers, materials, and warehouse tickets and cartons as inputs and is constrained by the cutting schedule and the size/color distribution. The main output of this activity is the manufactured garment.

Providing service to the customer is the other major function performed in an apparel enterprise and this is represented by the fourth box in the A0 diagram. It is responsible for interacting with the customer on various matters related to an order: confirmation, tracking, resolution of problems (delivery, quality, etc.). Since it interacts closely with the customer, it is responsible for issuing the final order to cut which serves as the input to the *Plan and Prepare for Manufacture* function. It also issues the shipping instructions for the finished goods. Another major function performed in an apparel enterprise is the distribution of the finished goods and this is represented by the fifth box in the A0 diagram. This function is constrained by the shipping instructions issued by the *Provide Customer Service* function.

Information about the various functions is vital for the efficient operation of the enterprise. Consequently, every function generates information on its performance (e.g., production data, quality problems) as one of its outputs. This information is monitored and utilized for setting standards and providing data for the operation of the enterprise. This function is represented by the sixth box in the A0 diagram. It is constrained by industry standards and practices in wages, engineering and quality.

The mechanisms have been shown only at the lowest level in the function hierarchy where they are relevant. Figure 4.4 through Figure 4.9 show the six functions on the A0 diagram in greater detail.

4.3.2 Zooming-in on the Manufacture Garment Function

The A0 diagram provides only a bird's-eye view of the major functions. Each of the functions in the A0 diagram has been hierarchically decomposed to expose greater details about that function.To illustrate this hierarchical modeling process, the decomposition of the *Manufacture Garment* function present on the A0 diagram is discussed. Figure 4.1 shows the part of the model described as a node tree diagram.

Figure 4.6 shows the decomposition of the *Manufacture Garment* function into the five functions that are carried out for manufacturing the garment. This is known as the A3 diagram since it corresponds to the third box in the A0 diagram. The top right- hand corner in the diagram provides the context and the level in the hierarchy. The inputs, constraints, mechanism and outputs in this diagram are inherited from the A0 diagram. Thus, the *Manufacture Garment* function in the A0 diagram can be thought of as the parent for the various functions in the A3 diagram. Note that the mechanism, QC personnel, is shown in the *Perform Quality Audit* function (the fifth box) since it (the function) has not been decomposed further.

Moving one level down in the hierarchy, the *Cut Fabric* function in the A3 diagram has been expanded into the component functions (or children) as shown in Figure 4.10. This is known as the A32 diagram since it corresponds to the second box in the A3 diagram. The input I1 (*Marker & Materials from Warehouse*) in the A3 diagram has been broken down into *Fabric*, *Marker*, *Trim* and *Accessories* in the A32 diagram thus providing more information on the process. Also, the outputs *Cutting Report* and *Complete Cut Package* in the A32 diagram correspond to the outputs of the *Cut Fabric* function in the A3 di-

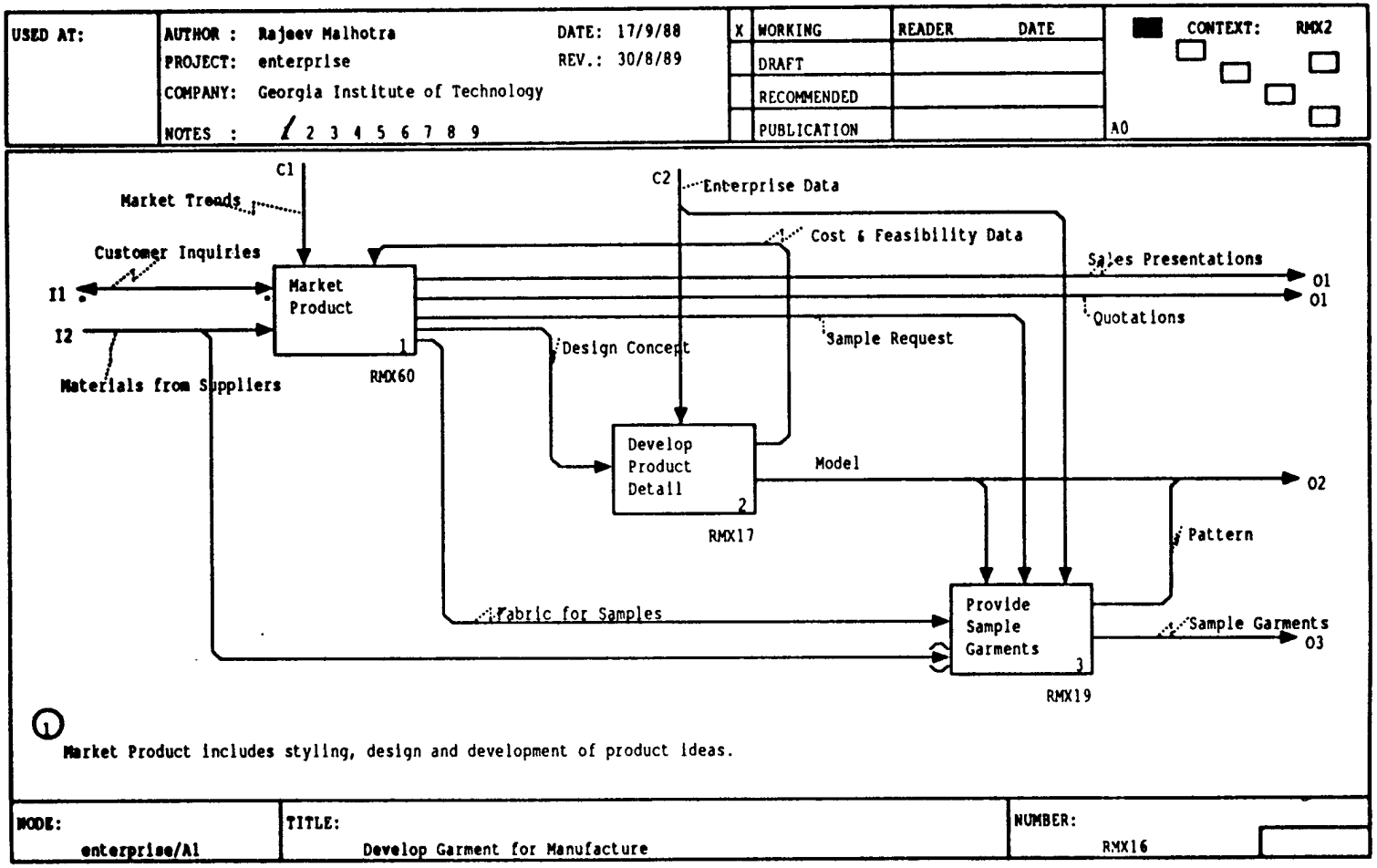


Figure 4.4. The *Develop Garment for Manufacture* Function

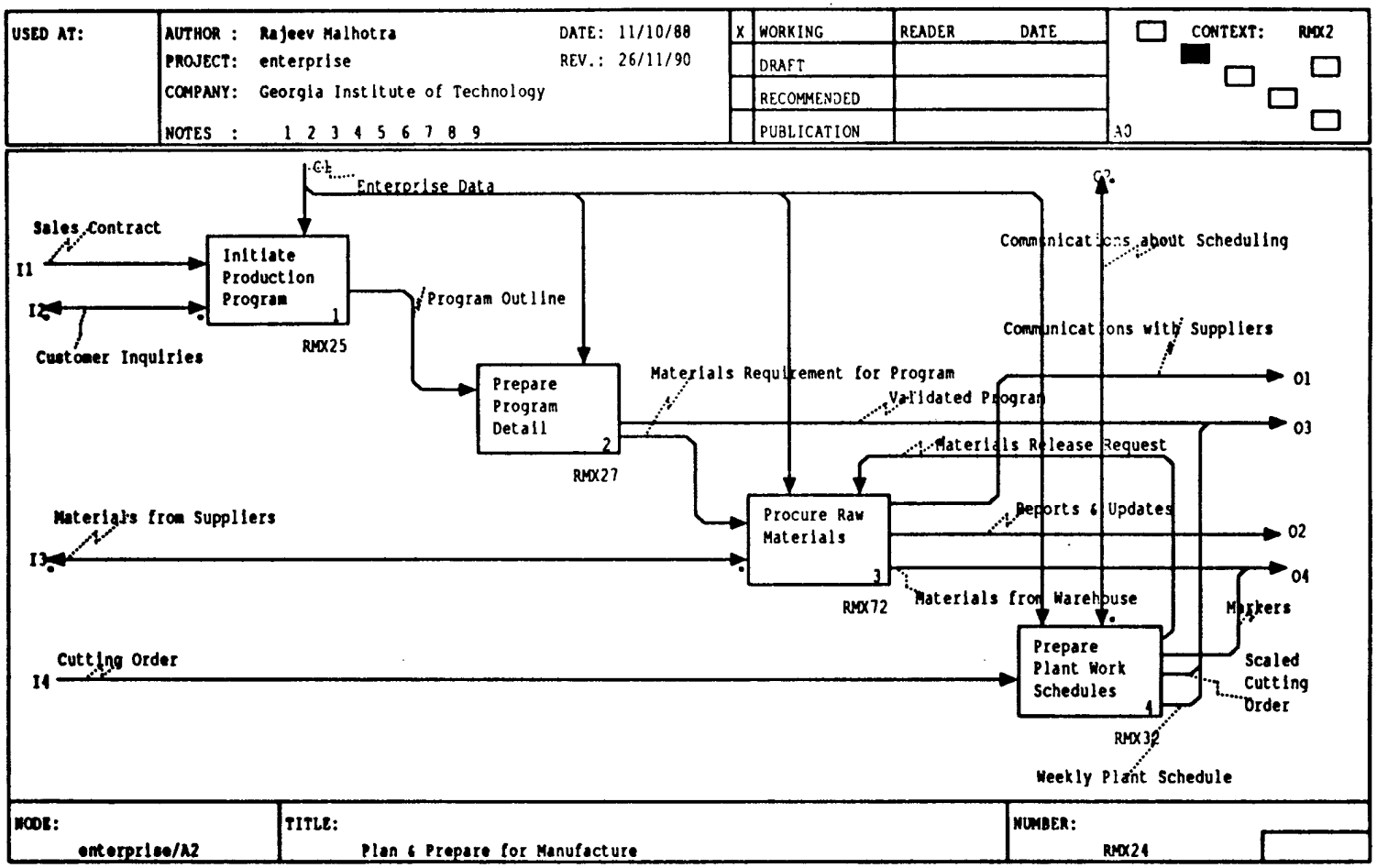


Figure 4.5. The *Plan & Prepare for Manufacture* Function

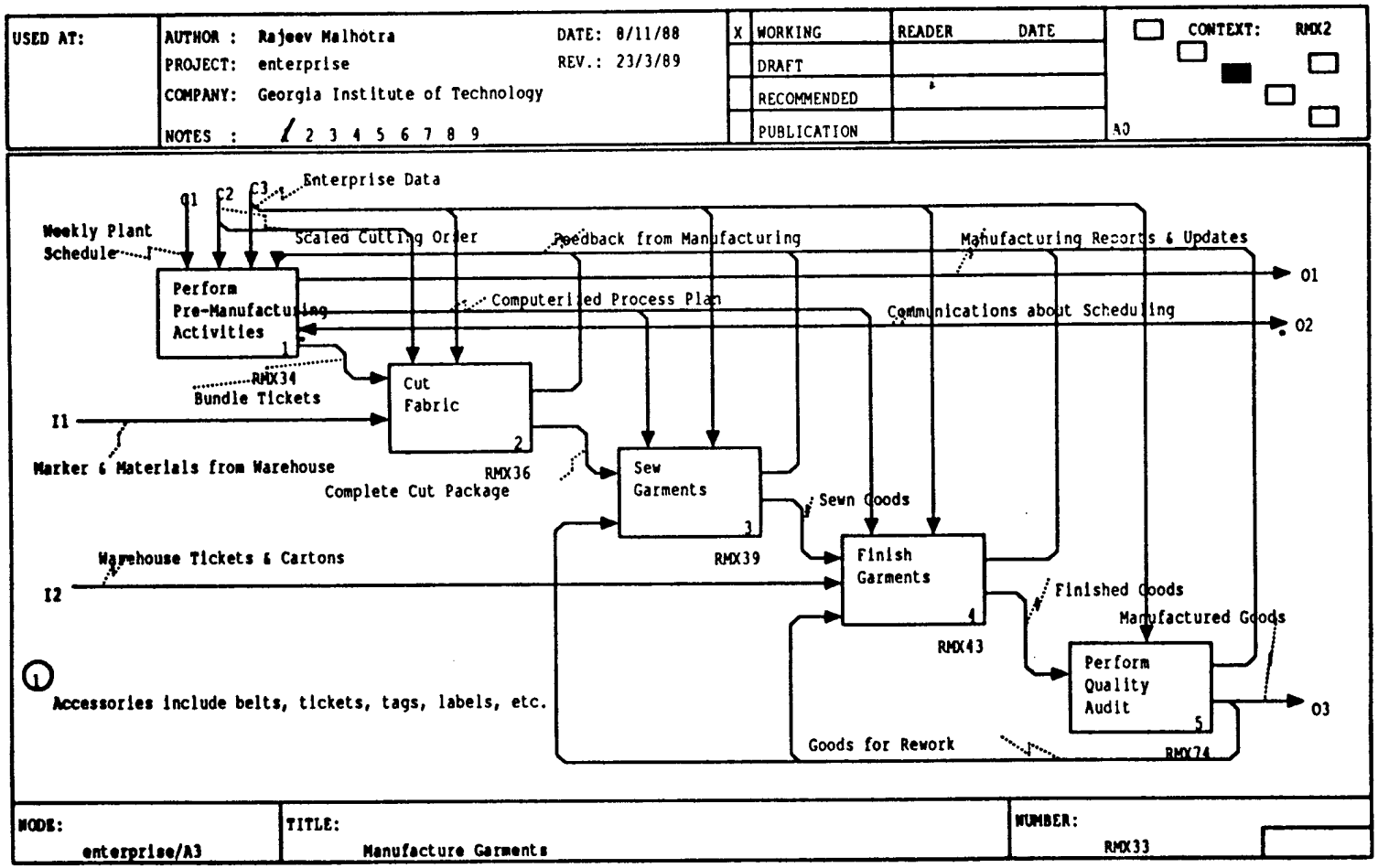


Figure 4.6. The *Manufacture Garment* Function

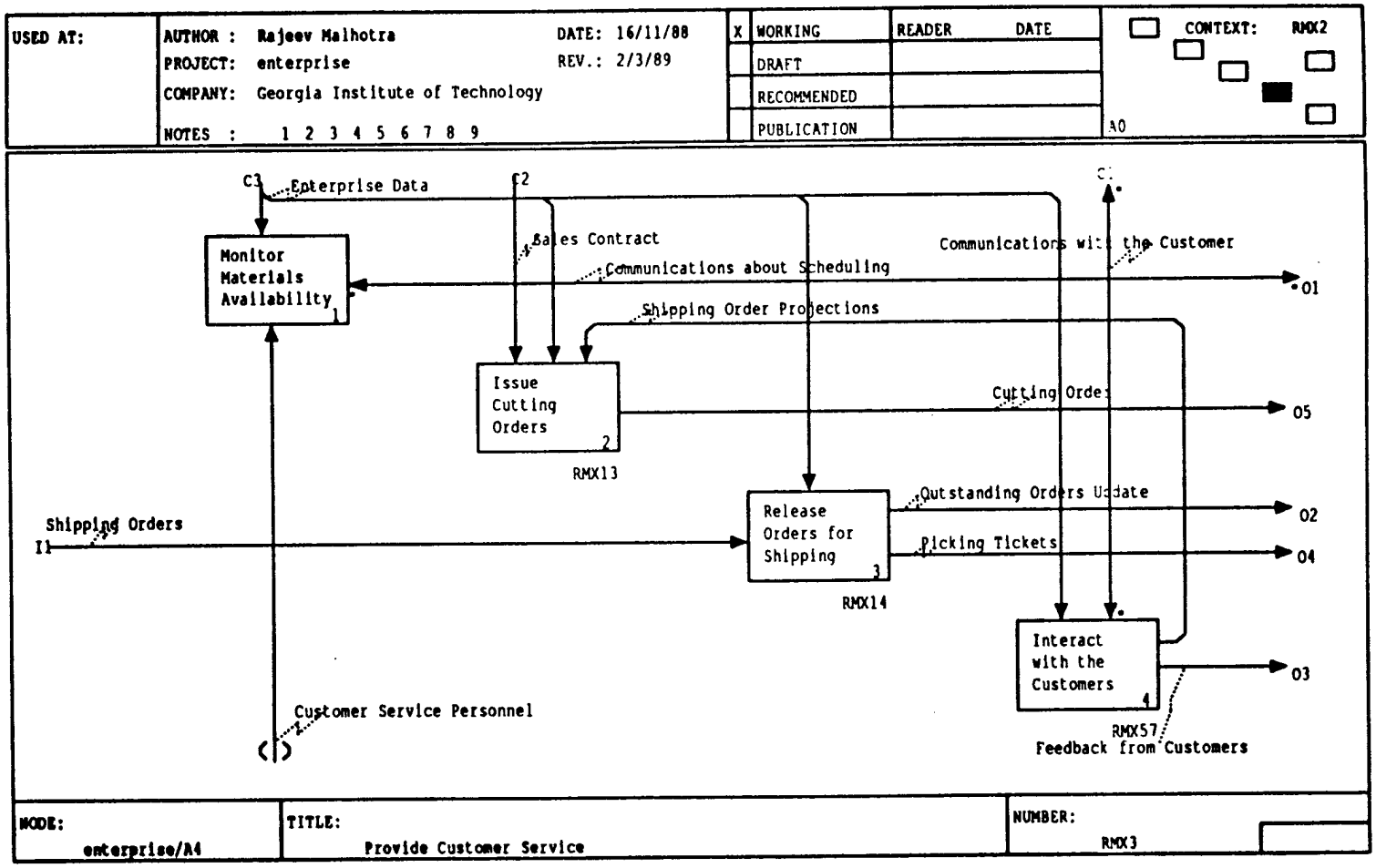


Figure 4.7. The *Provide Customer Service* Function

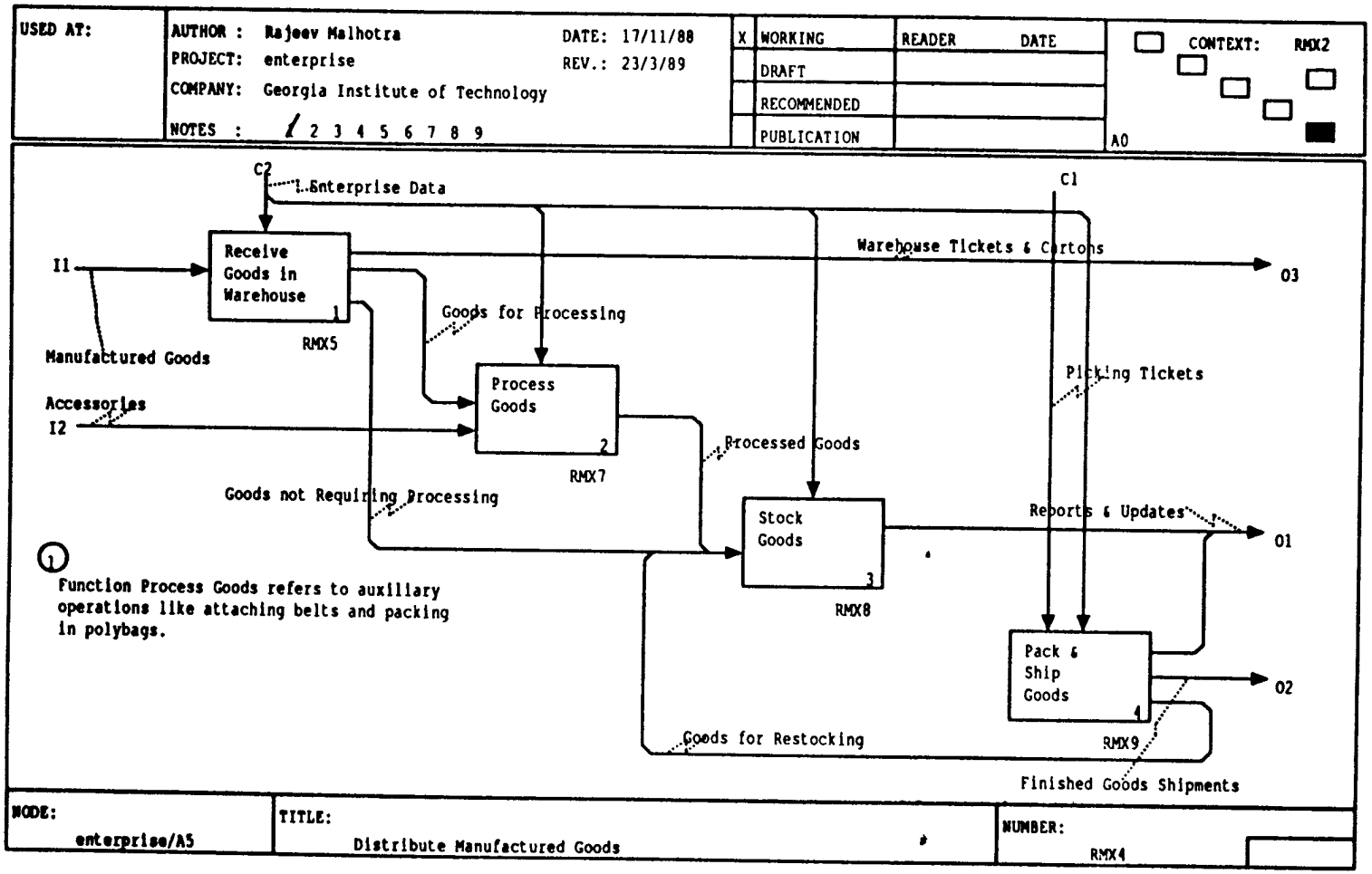


Figure 4.8. The *Distribute Manufactured Goods* Function

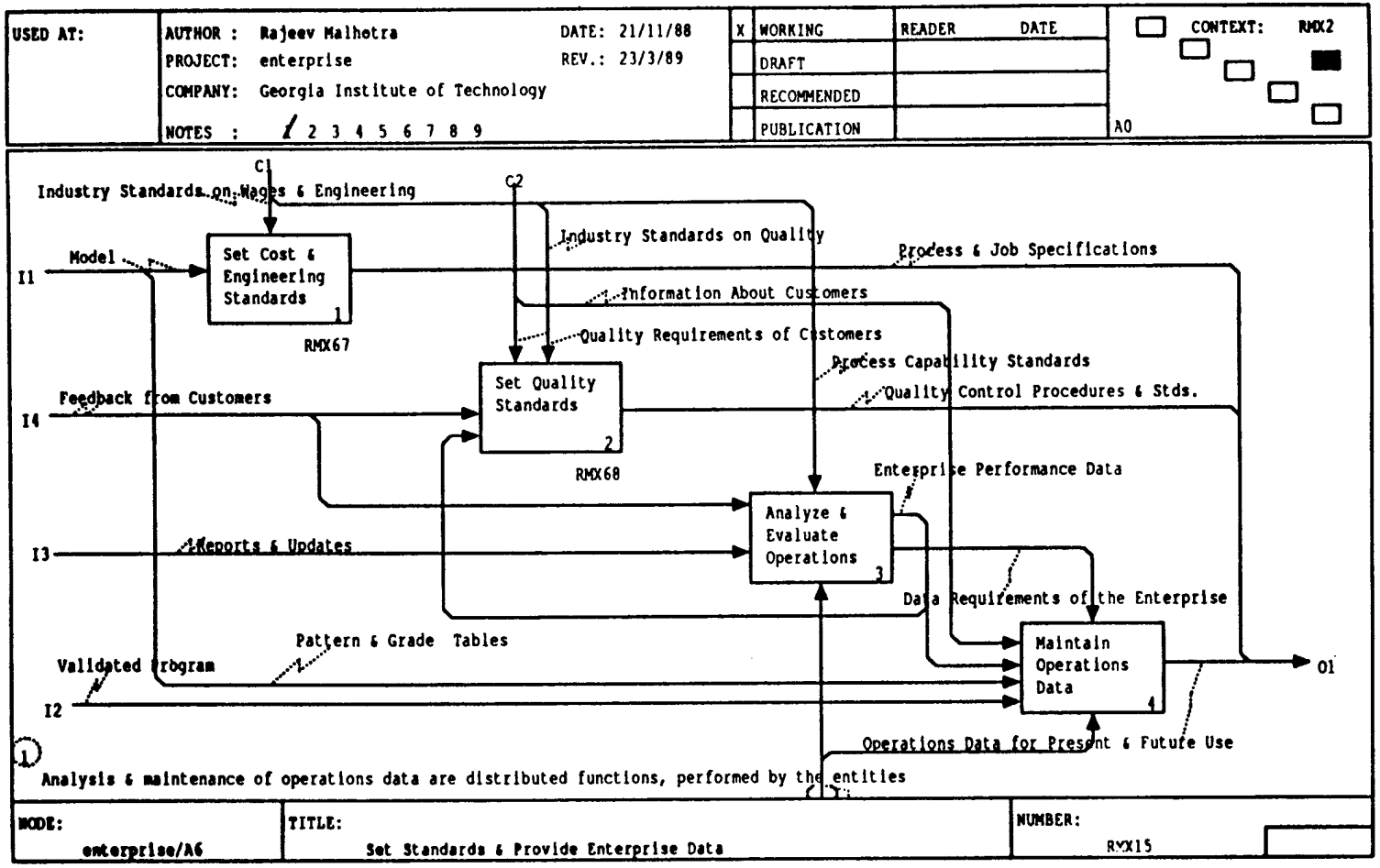


Figure 4.9. The *Set Standards & Provide Enterprise Data* Function

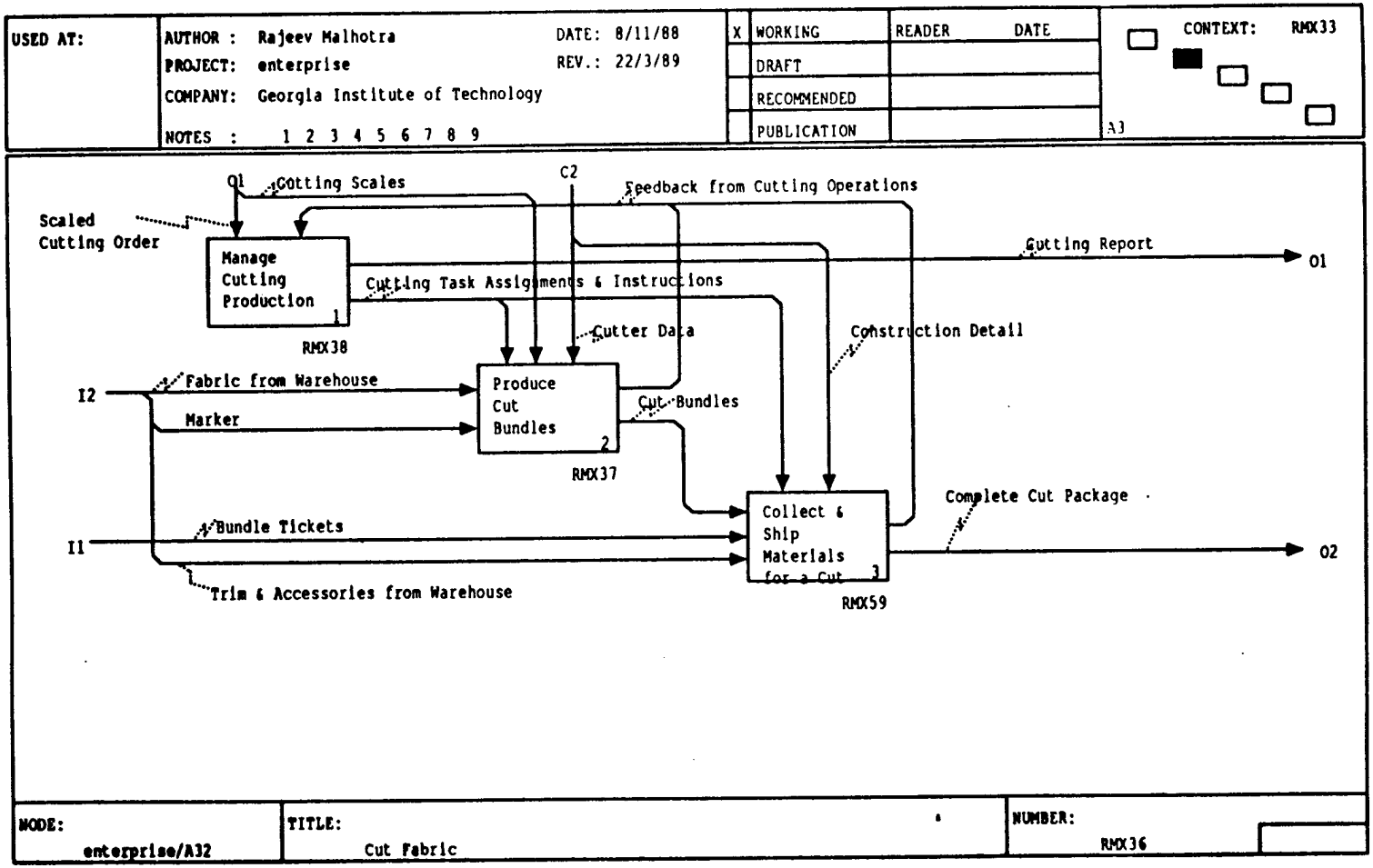


Figure 4.10. The *Cut Fabric* Function

agram. The same is true of the constraints C1 and C2.

Figure 4.11 shows the expansion of the *Produce Cut Bundles* function into its component functions. This is known as the A322 diagram since it corresponds to the second box in the A32 diagram. Likewise, the further expansion of the *Spread Fabric* function is shown in Figure 4.12 and this is the A3221 diagram. Finally, the expansion of the *Stop & Remove Defects* function is shown in the A32212 diagram in Figure 4.13. This diagram clearly shows the various functions performed in removing defects in the fabric during the spreading operation. Thus, the modeling process involves hierarchically breaking down the higher level functions into lower and lower levels until the desired degree of detail has been achieved.

Similar decompositions of all the major functions have been carried out to the desired degree of detail resulting in the AS IS function architecture for the apparel manufacturing enterprise. The model presented here incorporates numerous revisions and refinements that resulted from the iterative review process. Terminology that was specific to the enterprise that served as the source for model data, has been generalized. The initial version of the model also revealed many obviously redundant functions in the representative enterprise. Such redundancies have also been eliminated from the model as part of the review and refinement process. A more thorough analysis of the model has been left to the TO BE architecture development stage.

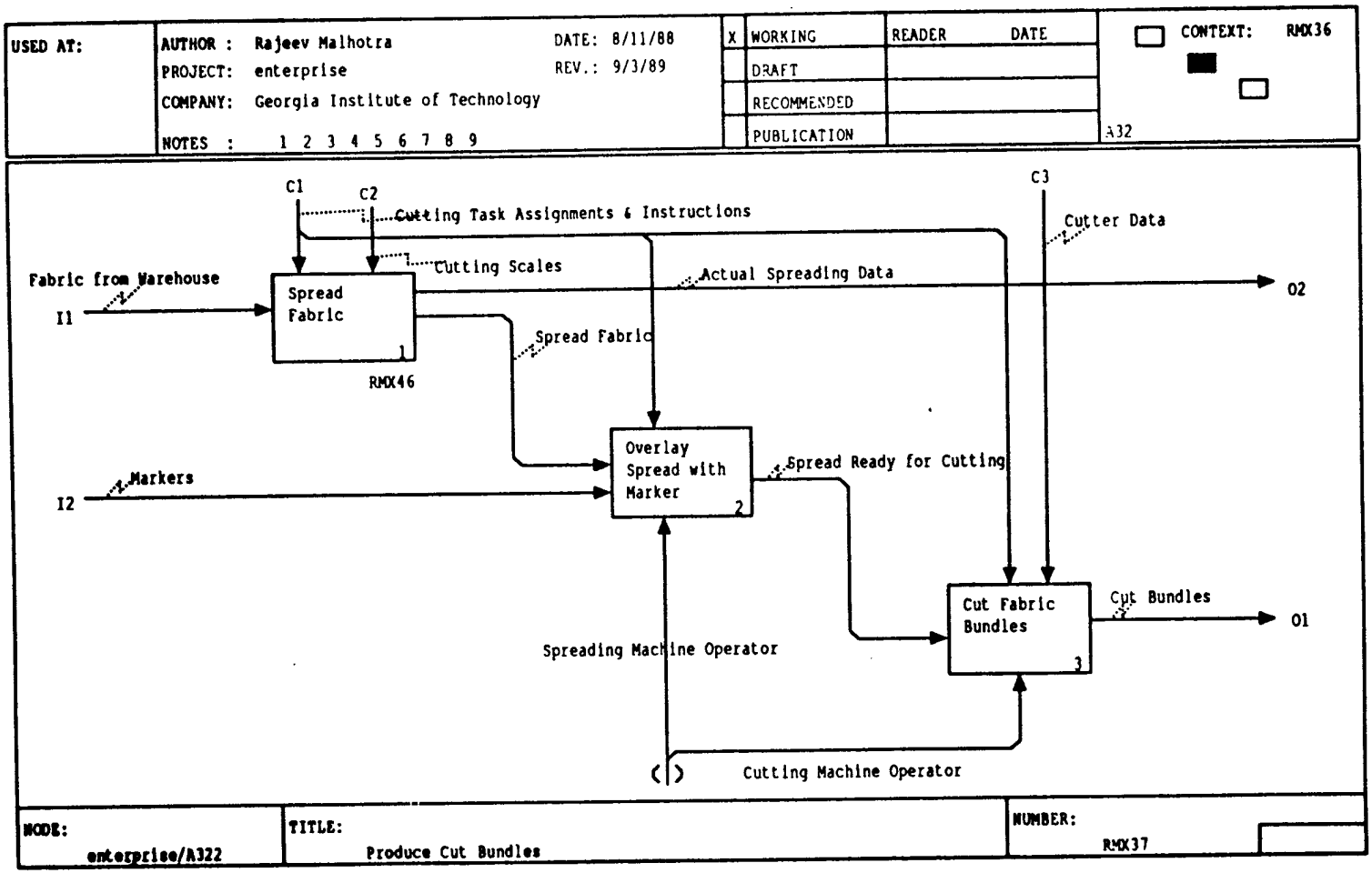


Figure 4.11. The *Produce Cut Bundles* Function

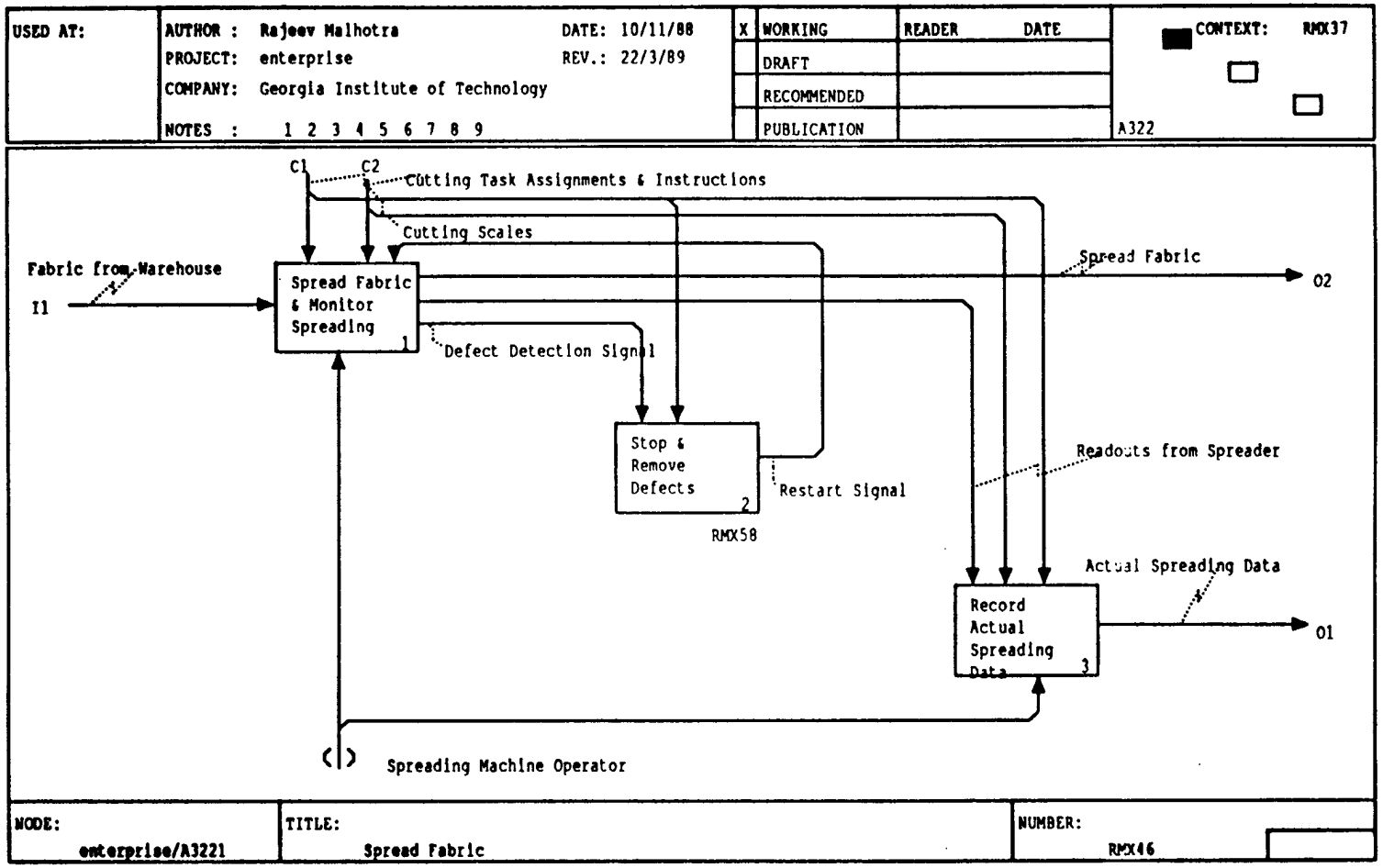


Figure 4.12. The *Spread Fabric* Function

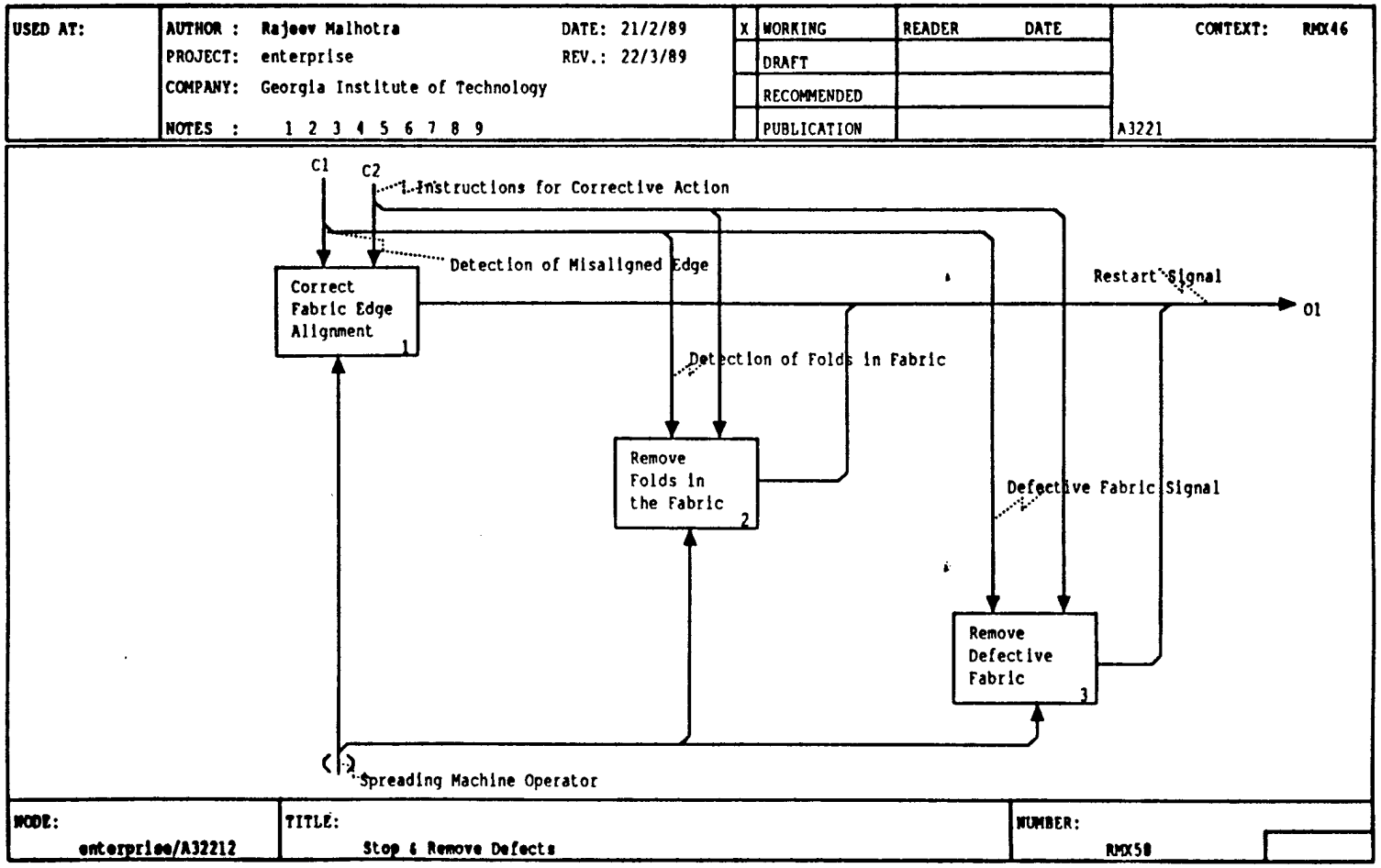


Figure 4.13. The *Stop & Remove Defects* Function

CHAPTER V

IFEM: AN INTEGRATED FRAMEWORK FOR ENTERPRISE MODELING

When an architecture for a large and complex system, such as an apparel enterprise is developed, the modeling task is facilitated if the architecture is resolved into multiple models each one of which focuses on one particular aspect of the system. However, since the resulting models represent different but complementary views of the same system, it is important to ensure that the models represent exactly the same domain and are consistent with each other. As the size and the complexity of the models increase, it becomes exceedingly difficult to ensure consistency without well-defined methods. The IDEF and other modeling methodologies reviewed in Chapter II do not provide any in-built means for maintaining consistency between models, but leave this task to the modeler. Therefore, as part of this research, a modeling framework which extends the IDEF methodology to address the consistency issue, was developed. The proposed integrated framework for enterprise modeling (IFEM) also overcomes the shortcomings of the $IDEF_2$ dynamics modeling methodology discussed in Chapter II. The TO BE architecture of the apparel manufacturing enterprise was developed using IFEM.

5.1 IFEM Concepts

The key idea underlying IFEM is to maintain consistency between the function, information and dynamics models of an enterprise by defining the modeling elements common to the models once and sharing them between the three models. Thus, in IFEM, the function, information and dynamics models are integrated into a single cohesive architecture of the enterprise.

A precise description of how the enterprise (being modeled) functions is captured in the IFEM function, information and dynamics models of the enterprise. Each model plays a specific role in the enterprise architecture developed using IFEM. Through the functional breakdown of the enterprise provided by the IFEM function model, the task of modeling how the enterprise functions is resolved into modeling how individual functional components work and interact with each other. Within the context of the IFEM function model, the IFEM dynamics model describes how individual functional components work and interact with each other dynamically. The structure of the data that is maintained to support the functions of the enterprise, and the relationships between the entities that the data represents are defined in the IFEM information model. These entities, such as fabric, trousers, purchase orders, schedules, etc., are processed by the functions of the enterprise and are represented in the IFEM function model as interfaces to the functions. Thus, functions and the entities processed by the functions are the elements that are shared among the three models in IFEM.

5.2 IFEM Function Modeling Methodology

The $IDEF_0$ function modeling methodology was selected for developing IFEM function models. Some of the strengths of $IDEF_0$, that make it a suitable methodology for function modeling, were discussed in Chapter II. $IDEF_0$'s hierarchical cell modeling technique provides a powerful and well-defined notation for representing the functional structure of an enterprise at a desired level of detail. The inputs, outputs, constraints and mechanisms (ICOMs) explicitly model interfaces to each function and inter-connections between functions; there are no hidden interactions between functions. Thus, the boundaries of each function are clearly demarcated in an $IDEF_0$ function model, making it a suitable foundation on which IFEM architecture of an enterprise could be built.

5.3 IFEM Information Modeling Methodology

The choice of IDEF$_1$x as the information methodology for TO BE architecture was discussed in Chapter II. An IDEF$_1$x model of an enterprise provides a conceptual view of the enterprise data, based on which an integrated relational database to support CIM in the enterprise could be designed. The information model defines the structure of the data that is an abstract representation of the entities that are modeled as ICOMs in the IDEF$_0$ function model. In IFEM, the entity definitions are shared between the information and function models by defining the ICOM interfaces in the function model in terms of the IDEF$_1$x entities.

5.3.1 Limitations of the IDEF$_1$x Model

The IDEF$_1$x data structures cannot be directly assigned to ICOMs because the application of rules of normalization [Codd70] to the IDEF$_1$x model results in fragmentation of real-world entities (that ICOMs represent) into multiple IDEF$_1$x entities. For example, the attributes of a purchase order are structured in two IDEF$_1$x entities - *purchase order* and *purchase order item*. Therefore, the normalized IDEF$_1$x entities do not individually represent ICOMs. The entities that ICOMs represent have to be composed from multiple IDEF$_1$x entities. As part of IFEM, an extension to the IDEF1x information model is proposed whereby the real-world entities are defined as composites of related IDEF$_1$x entities. The proposed extension provides the means for closely integrating the function and information models.

5.3.2 Extensions to the IDEF$_1$x Model

The composite structures that represent ICOMs in IFEM are called object *classes* (Figure 5.1). A particular instance of a class, e.g., purchase order # 101, is called an *object*. The characteristics of the real-world entities that ICOMs represent, are modeled as *features* of these object classes. Features are a more general case of attributes that represent the char-

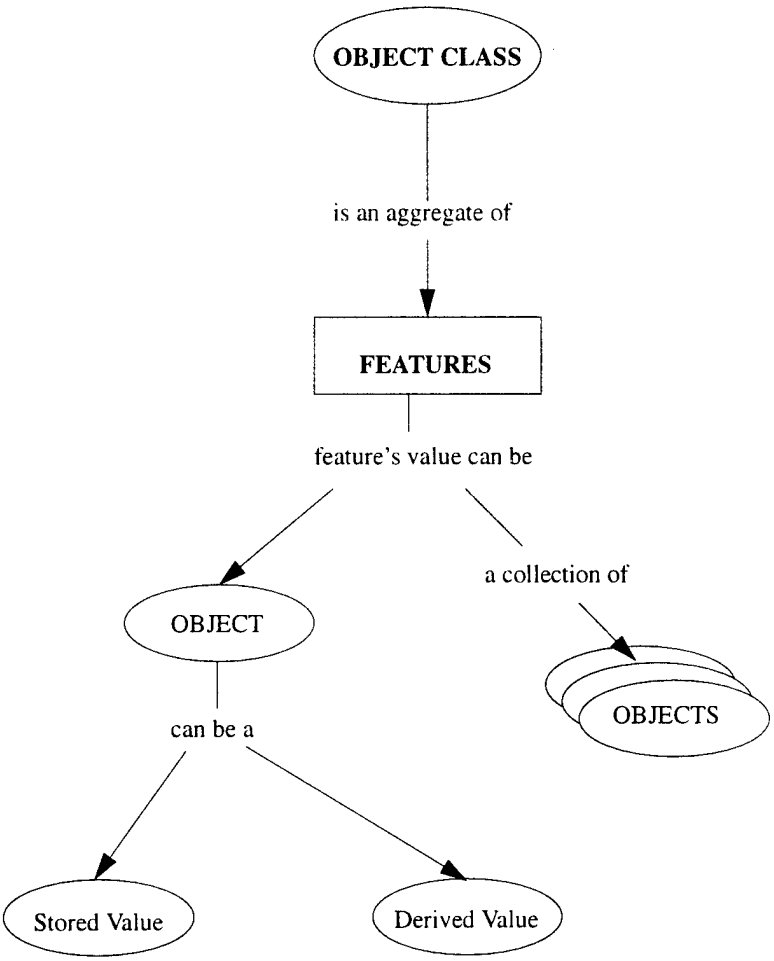


Figure 5.1. Structure of an IFEM Object Class

acteristics of the entities in IDEF$_1$x models. IDEF$_1$x attributes represent only the fundamental characteristics of the entities. Fundamental characteristics of an entity are those characteristic that cannot be derived from other characteristics. For example, length and width are fundamental characteristics of a room whereas its area, which is expressed as the product of length and width, is a derived characteristic. The features in IFEM represent both, fundamental and derived attributes, as the latter are often of interest in the model. The value of a derived feature is not a stored data item, but is computed by a formula or a procedure from the values of other features.

Often, a characteristic of interest in a modeled entity has a set of data items as its value instead of a single data item allowed in IDEF$_1$x. For example, one of the characteristics of a garment pattern is the list of parts that make up the pattern. In IDEF$_1$x, a dependent child entity has to be created to model multi-valued attributes to conform to the first normal relational form. In the current example of garment pattern, the actual pattern is represented by two IDEF$_1$x entities - *pattern* and *pattern part*. *Pattern part* is related to the *pattern* to which it belongs by making the key attributes of *pattern* the foreign key of *pattern part*. To represent an actual pattern, data items from both the entities have to be aggregated. In IFEM all the data items that represent an ICOM are aggregated into a single object by allowing the features of an object to be multi-valued.

In IFEM, the value of a feature of an object can be another object. For example, a pattern, which describes the shape of a garment style, can be a feature of the garment style. By allowing features to have objects as their values, a hierarchy of object classes can be built in which complex object classes are defined as composites of simpler object classes. The value assigned to an object-valued feature is a single value that uniquely identifies the object being referenced by the feature. This value, called the *object identifier*, provides a simple means, independent of the primary key attributes, for expressing relationships between objects.

5.3.3 Role of Information Model in IFEM

The functions of the enterprise can be viewed as applications that reference or manipulate the data maintained in the enterprise database (the functions that physically transformation entities, e.g., cut fabric, move garments from storage to packing area, etc., also transform the data entities that are abstract representations of the corresponding physical entities). Whereas, the IDEF$_1$x model describes how the data should be structured in the enterprise database, the object model of IFEM describes how this data would be viewed by the functions. Thus the IFEM *object model* defines the interfaces between the functions and the enterprise database. If the database is implemented using relational technology, these interfaces will have to be implemented by the application software that controls the functions, because relational database management systems do not support the IFEM object model concepts, such as multi- and object-valued attributes as part of the schema definition. These concepts are supported in the still-evolving object-oriented database technology. Once the object-oriented databases become commercially acceptable, the IDEF$_1$x model and the IFEM object model could be replaced by a single model that would provide an object-oriented database schema that would support the application interfaces.

5.4 IFEM Dynamics Modeling Methodology

The dynamics model provides an understanding of how the functional components of the system behave over time and how these components interact with each other to produce a flow of entities through the system. The dynamics model describes the dynamic aspects of the functions and the entities whose static description is contained in the function and information models. The model provides the means to analyze the behavior of the system through simulation.

5.4.1 Need for IFEM Dynamics Modeling Methodology

Although, a $IDEF_2$ dynamics model provides a complete description of the system for simulation, it is a stand-alone model that duplicates most of the information captured in the function and information models. Redefinition of functions and entities in $IDEF_2$ dynamics model makes its integration with the corresponding function and information models difficult.

Apart from the integration problem, the $IDEF_2$ methodology has other limitations that make the methodology unsuitable for IFEM dynamics modeling. The most serious limitation of the $IDEF_2$ methodology is that it models the dynamics behavior of the system as it pertains to the flow of a specific entity through the system. In modeling CIM systems, which are characterized by their ability to reconfigure dynamically to process a variety of entities, the $IDEF_2$ approach of modeling the flow of a single entity through the system is not suitable. The methodology does not provide the means for referencing resources, entities and process sequences as generic parameters. Hence, generic models, that are not restricted to the flow of a specific entity, are difficult to build using $IDEF_2$.

5.4.2 The IFEM Dynamics Modeling Approach

The shortcomings of the $IDEF_2$ dynamic methodology are addressed in IFEM by adopting a different approach to dynamic modeling than the one taken by $IDEF_2$. In IFEM, instead of modeling the flow of an entity through the system, the dynamic behavior of the functional components of the system, identified in the function model, and the dynamic interactions between these functions are modeled by extending the $IDEF_0$ function model. The resulting dynamics model is completely integrated with the function and information models, and captures the dynamics behavior of all the functions of the enterprise instead of only those that participate in the flow of a particular entity.

In IFEM, once the functional structure of the enterprise has been broken down to

a desired level of detail in the function model and the ICOMs defined in terms of the entities modeled in the information model, the dynamic description of the lowest level functions that are not decomposed further, is developed. The context of the function whose dynamic description is being developed, is provided by the ICOM interfaces of that function. The dynamic description strictly adheres to this context. A function has access to only those entities that are represented by its inputs and control and it can process only those entities that are represented by its outputs. Resource entities, such as operators and equipment, available to the function are represented by the mechanism interfaces to that function. The only interactions that a function is permitted with other functions are the ones represented explicitly by ICOMs.

Description of the Dynamics of a Function: The dynamics of a function is described in a script very similar to a subroutine in a computer programming language. The script contains a procedural description of how the entities represented by the inputs and controls of a function are transformed into entities represented by output, and how the resources represented by mechanisms are used to carry out this transformation. The ICOMs are analogous to parameters in subroutines. In the scripts, specific entities are not referenced; instead all the references to the entities are as parameters, making the description generic. The parameters are replaced by specific entity instances when a function is activated, just like the parameters in a subroutine are replaced by the actual values passed to the subroutine when it is called. To prevent hidden interactions, similar to side effects in subroutines, the functions are permitted to modify only the data items that represent the output entities. The data on input and control entities can only be referenced, but cannot be modified.

Consider an example of a model of a flexible machining system (FMS). One of the functions of a FMS is to load a part on a machine (Figure 5.2). The input to this function

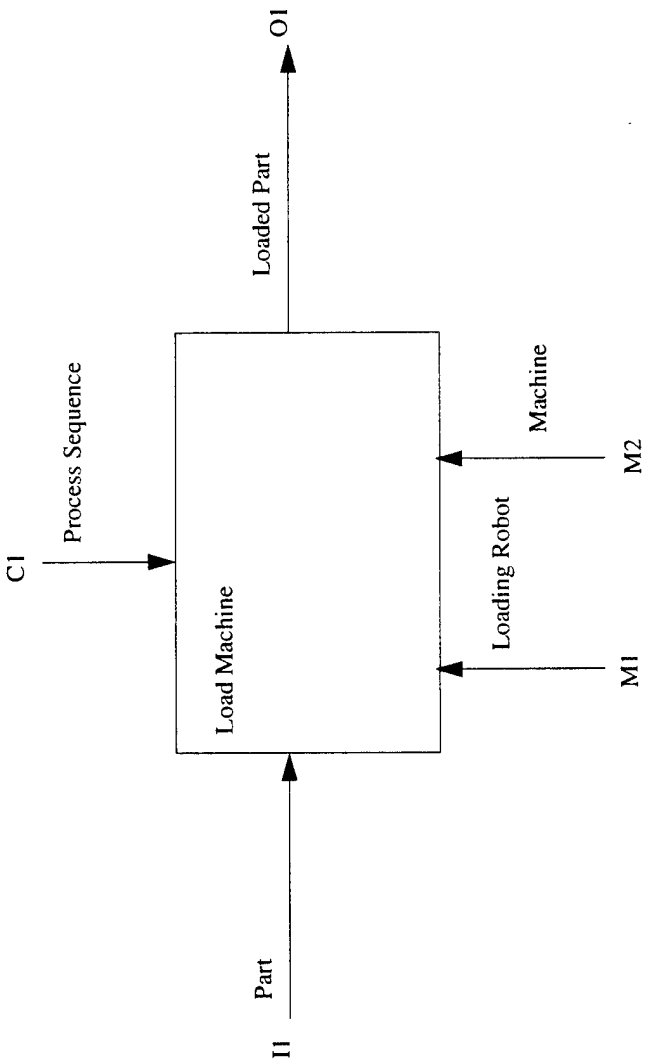


Figure 5.2. The *Load Machine* Function

is the part to be loaded (I1) and the output is a part loaded on the machine (O1). The resources used are a loading robot (M1) and a machine on which the part is loaded (M2). The function is controlled by the process sequence of the part (C1). A generic dynamics description of this function could be as follows:

1. Select a part from input I1.
2. Lookup the process sequence for the part from control C1 to determine on which machine should the part be loaded next.
3. Engage that machine from M2 when it becomes available.
4. Engage an available robot M1 to carry out the loading.
5. Load part I1 on machine M2.
6. Release the loading robot M1.
7. Queue up the part loaded on the machine as output O1.

This description is generic because it describes the *Load Machine* function without referring to a specific part, robot or machine. When the function is activated for loading a particular part, the selection of a specific robot and a machine are made based on the process sequence for the part. The feature that distinguishes this description from an equivalent $IDEF_2$ dynamic description is that the process sequence is not considered a part of the function's dynamic description, but is treated as a data input to it. By separating out the elements that are specific to a particular entity from the dynamic description of a function, the dynamics description is made generic in IFEM. The IFEM dynamic modeling approach is particularly well-suited for modeling flexible data-driven processes that are characteristic of CIM systems.

In the current FMS example, it is possible that two or more parts are being processed in the system at any given time and many machines get loaded simultaneously. In $IDEF_2$, this situation would be modeled by having one node for each loading station in the system. In IFEM, this situation is treated as multiple concurrent activations of the same

function *Load Machine*, instead of being modeled as multiple nodes. Number of simultaneous activations depends on the availability of resources and other conditions. For example, function Load Machine may be activated if a loading robot is available and a part is waiting to be loaded. In IFEM the conditions that trigger the activation of a function are specified as a part of the dynamics model. Treating multiple processes as activations of a single function permits development of models that are independent of actual size of the system. For example, the dynamics model of an FMS is not altered if an extra loading robot, or an extra machine is added to the system configuration. Thus, IFEM dynamic models are generic with respect to the scale of the system being modeled.

Dynamic Interactions between Functions: In IFEM, to model dynamic interactions between the functions, the ICOMs interfaces between functions are assigned the following dynamic attributes:

1. Queue
2. Signal

Using the *queue* attribute, the ICOMs are treated as channels through which entities flow between functions. A function queues up the processed entities at its output interfaces. The entities wait in the ICOM queues till they are retrieved by the functions at the other end for further processing. A maximum capacity can be specified for an ICOM queue: when an ICOM queue gets filled to its capacity, the function (for which the ICOM is an output) is halted till sufficient space is once again available in the queue.

The *signal* attribute provides the means for communication between interacting functions. A signal for an ICOM is raised by a function to indicate that an entity has been queued at the ICOM interface by the function. This signal is received by the functions at the other end to detect the availability of entities in the ICOM queue. These signals may also trigger an activation of a function if they are part of the trigger condition.

5.4.3 Syntax for IFEM Dynamics Model

In IFEM, the dynamic behavior of the functions is modeled using primitives similar to the ones used in simulation modeling [ICAM81b, Pegden82]. These primitives represent various dynamic actions that take place when functions are activated. These actions include retrieval of entities from input queues, lookup of input or control data, delays representing processing times, engaging and disengaging of resources, queueing of output entities and raising of signals associated with outputs. The sequence of actions performed by a function are captured in a *script* associated with that function, using the dynamics modeling primitives. The syntax of these IFEM primitives is discussed in detail in Chapter VIII.

CHAPTER VI

TO-BE ARCHITECTURE: THE INFORMATION MODEL

One of the main objectives of CIM is to integrate the various functional components of a discrete-part manufacturing enterprise into a unified system whose functions can be readily monitored and controlled. This objective is achieved by linking all the functional components of the enterprise to a common information system through which the data generated by a function is presented in a meaningful way to other functions that either monitor this function or are controlled by it. Data, structured and presented in a meaningful form, is called information. Information can be shared by functions only if it means the same to all the functions that share it. The information model provides a single consistent definition of data based on the semantics of the real world entities the data represents, and unbiased towards any single function. An information model for the apparel manufacturing enterprise is presented in this chapter. The model serves as a conceptual schema for an integrated information system that is the foundation of a CIM system for an apparel enterprise.

6.1 Model Syntax and Diagramming Conventions

The TO BE information model was developed using the IDEF$_1$x methodology and consists of a set of diagrams and an accompanying glossary. The diagrams depict the information structure in a graphical form as a map of data entities and their relationships, and the glossary provides a textual description of the data entities. The various model elements are represented graphically using the following diagramming conventions:

6.1.1 Entity

Entities are represented by rectangular boxes in the IDEF1x diagrams. Each entity

is given a unique name and number that appears at the top of the box. For example, the box representing the entity *pattern* is labeled E14/PATTERN (Figure 6.1).

6.1.2 Relationship

The relationships between entities are represented by directed lines joining the related entities. For example, the relationship "A pattern consists of many parts" is expressed as a line directed from E14/PATTERN to E15/PATTERN_PART (Figure 6.1). The dot at the end of the line is used to indicate the direction and letter *n* next to it represents the cardinality of the relationship, which is *one-to-many* in this case (i.e., a pattern consists of many parts whereas each part belongs to only one pattern). A parent-child relationship is said to exist between pattern and pattern part in which pattern is the parent and pattern part is the child.

6.1.3 Attributes

The characteristics that describe the entities are represented as attribute names and are listed inside the entity boxes. The attributes listed above the horizontal line dividing the entity boxes make up the primary key of the entity. The primary key uniquely identifies an instance of an entity. For example, each pattern is uniquely identified by its primary key consisting of attributes *BasPatNo* and *RunNo*. Attributes listed below the line are called non-key attributes. When a relationship exists between two entities, the primary key of the parent is inherited by the child as a foreign key. A foreign key is denoted by FK in parentheses after the inherited attribute name. For example, the attribute, *BasPatNo*, which is a part of the primary key of the entity, *pattern*, is inherited from its parent entity, E13/ BASE_PATTERN.

6.1.4 Independent vs. Dependent Entities

For an instance of an entity to exist, it must have values for all its primary key at-

tributes. For example, a base pattern's record cannot be maintained in the enterprise without assigning a value for its BasPatNo. Therefore, a base pattern does not exist on the records if it does not have a BasPatNo. If the primary key of an entity is inherited, as in the case of E14/PATTERN (Figure 6.1), then the existence of such entity depends on the existence of its parent(s). For example, a pattern that belongs to a base pattern type identified by BasPatNo 231 cannot exist in the records if there is no record for a base pattern with BasPatNo 231. A dependent entity is represented as a box with rounded corners and a dependence relationship is represented by a solid line.

6.1.5 Categories

Some information entities are related by their similarity to each other. For example, construction materials, such as trim, labels, thread, etc. are identified by a material code and a color code, but each has certain characteristics that are unique to it. For example, count may be an important characteristic of thread but it has no meaning for labels. The relationship among such entities is called categorization relationship and is illustrated for entity E34/MAT_VARIANT in Figure 6.2.

6.1.6 Functional Views

While modeling a complex system such as an apparel manufacturing enterprise, it is difficult to map all the relationships on a single diagram. For ease of modeling, the model is broken up into functional views each of which represents a particular aspect of the enterprise operation. These views remain parts of the same model because they share common entity definitions. Each functional view can be spread over multiple diagrams that are connected through *pageconnectors*. The node number and the title of the functional view are printed at the bottom of each diagram. For example, the two diagrams that make up the function view F200 entitled *Material Description* are shown in Figure 6.2.

The information model has been normalized to the third normal relational form

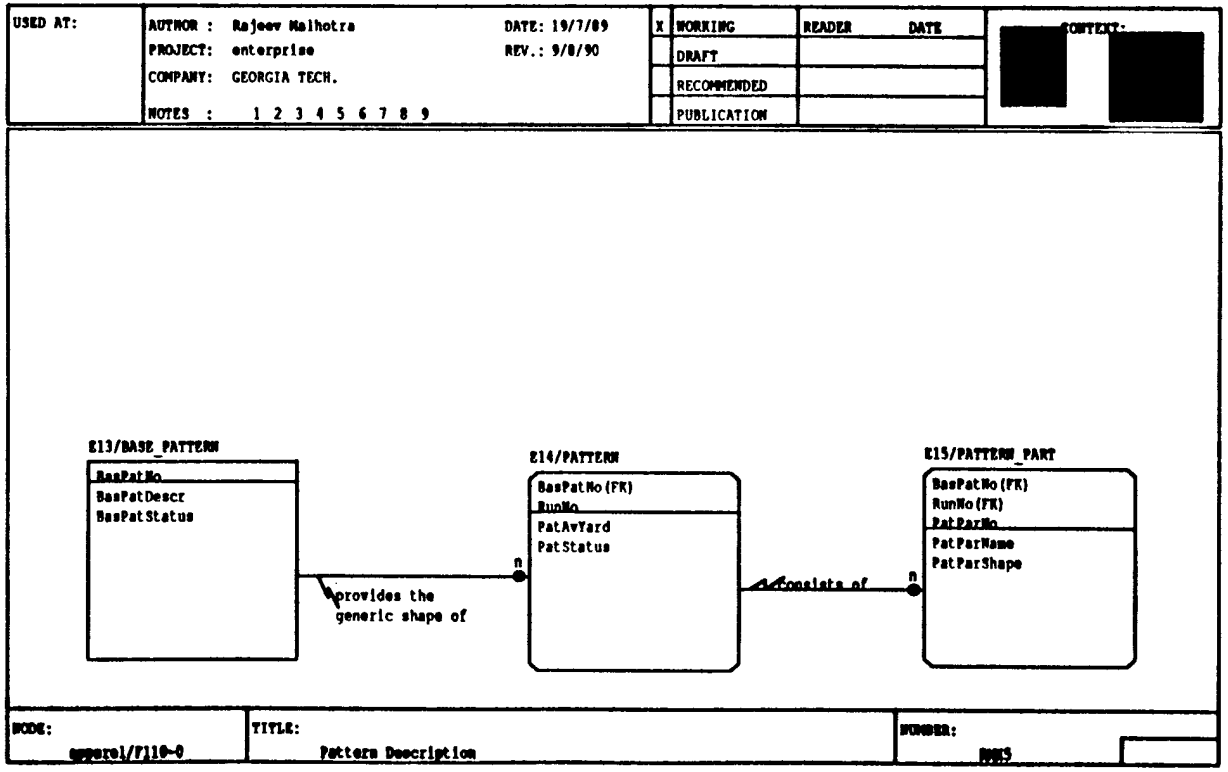


Figure 6.1. IDEF$_1$x Entity-Relationship Diagram

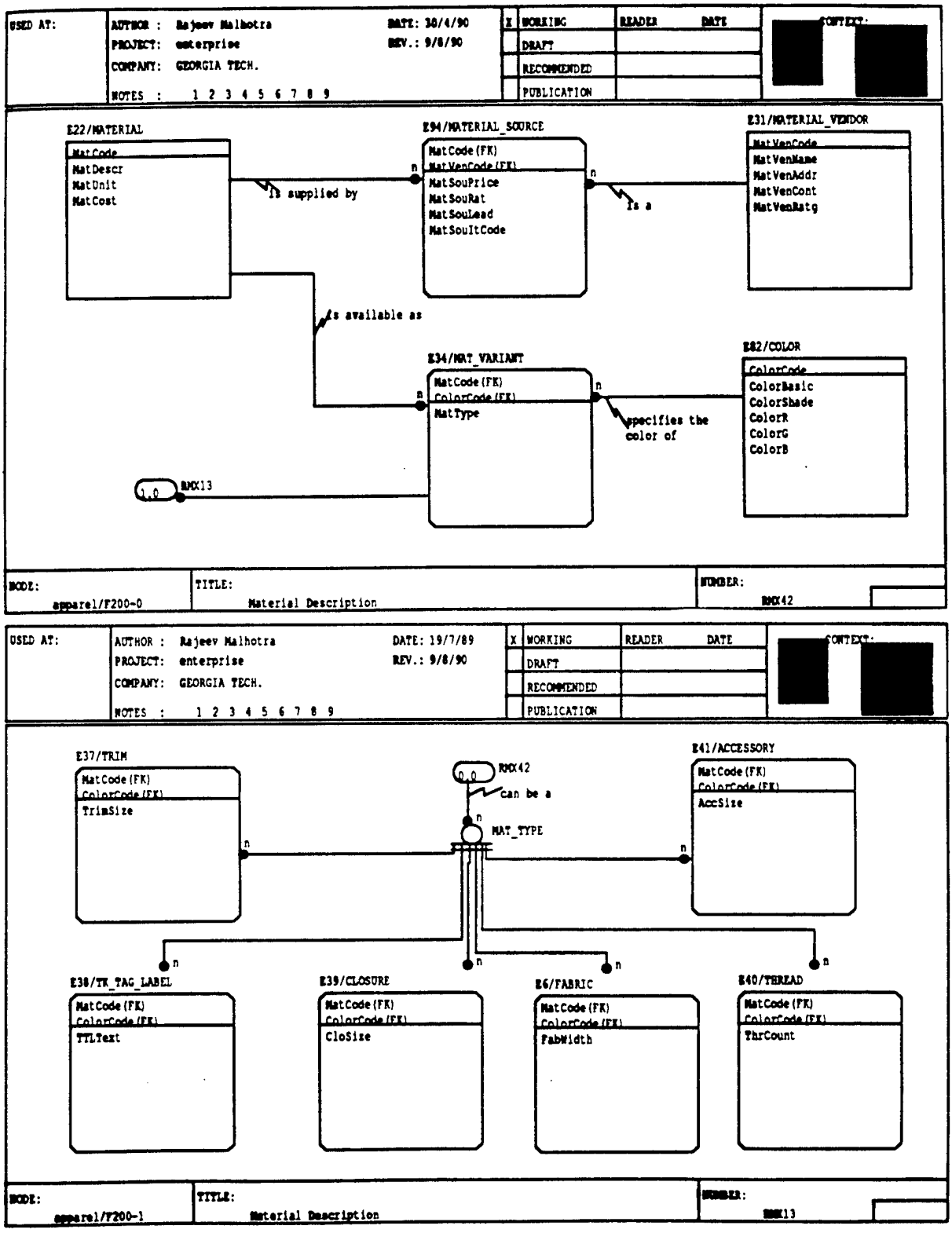


Figure 6.2. The Function View *Material Description*

[Codd70]. In this form, all the many-to-many relationships (e.g., many materials may be ordered on a purchase order and a material may be ordered through many purchase orders) have been transformed into one-to-many relationships by creating intermediate child entities between entities that have many-to-many relationships, and the non-key attributes of each entity are dependent only on the key attributes of that entity and nothing else. All the entities and their attributes have been defined in the dictionary that accompanies the model.

6.2 The TO BE Information Model

The TO BE information model defines the structure of the entities generated and processed by the functions of the apparel manufacturing enterprise and the relationships that exist between these entities. Since the model is a flat network of entities and their relationships that is too big to be presented as a single diagram, it has been broken down into functional views in the following areas:

1. Marketing and Product Development
2. Enterprise Support Services
3. Planning and Preparation for Production
4. Production Control
5. Manufacturing
6. Distribution

The TO BE information model spans all the enterprise functions that fall within the defined scope of the architecture and are represented in the AS IS function model. The entity definitions and relationships presented in the TO BE information model are derived from the glossary of the AS IS function model. In this section, the TO BE information model is discussed with the help of illustrative examples from the various functional areas covered in the model. The complete model, along with its glossary is presented in Appendix A.

6.2.1 Marketing and Product Development

The Marketing and Product Development function involves marketing of product ideas to customer and development of garment style. The result is an unambiguous description of what the garment looks like and how it is to be made. A style is developed for a customer from the style concept (a sketch, garment sample, etc.) which describes the customer's style requirements. Figure 6.3 shows the function view *Product Development and Description* which depicts the structure of the entity *style* and its relationships to its constituent entities, i.e., construction detail, pattern, fit and process plan. The view also depicts the relationship between a style and a style concept.

The structure of construction detail, pattern, fit and process plan are defined in separate functional views. For example, the structure of a construction detail is defined in the function view *Construction Detail* shown in Figure 6.4. Construction detail provides the description of garment features, such as front pockets, waistband, base, etc. The construction detail also specifies the types of materials to be used for constructing the garment features and instructions regarding their placement. It serves as a bill of materials for the garment. It is a list of feature items selected from a library of features. Materials for each feature item are specified while describing the item. Only the type of material is specified during style development stage since the color of the materials is different depending on the fabric used to make the garment. Colors for each material are specified at a later stage for each fabric to be used when the fabric colors are known. The manufacturing cost of the garment is determined by summing up the cost of materials used and the cost of operations associated with each garment feature.

6.2.2 Enterprise Support Services

The following support service functions fall within the purview of this model: Vendor development, industrial engineering and quality control. The information generat-

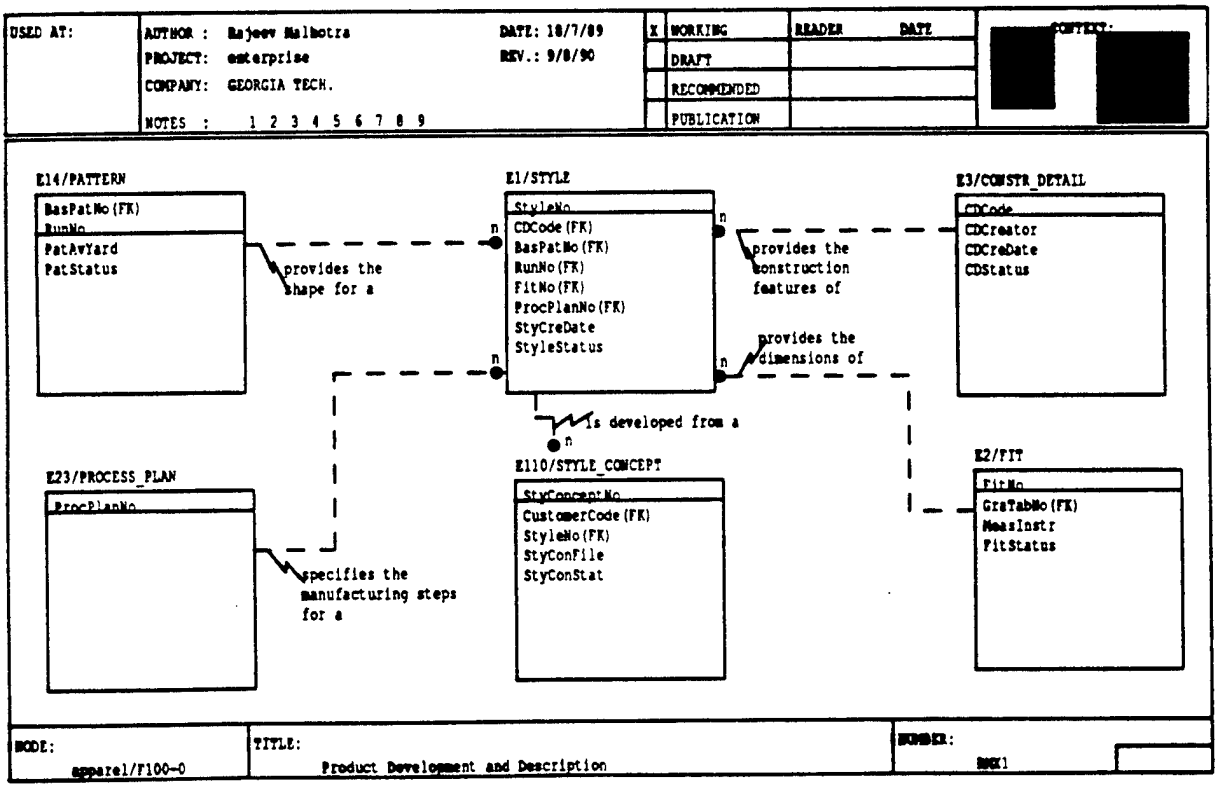


Figure 6.3. The Function View *Product Development and Description*

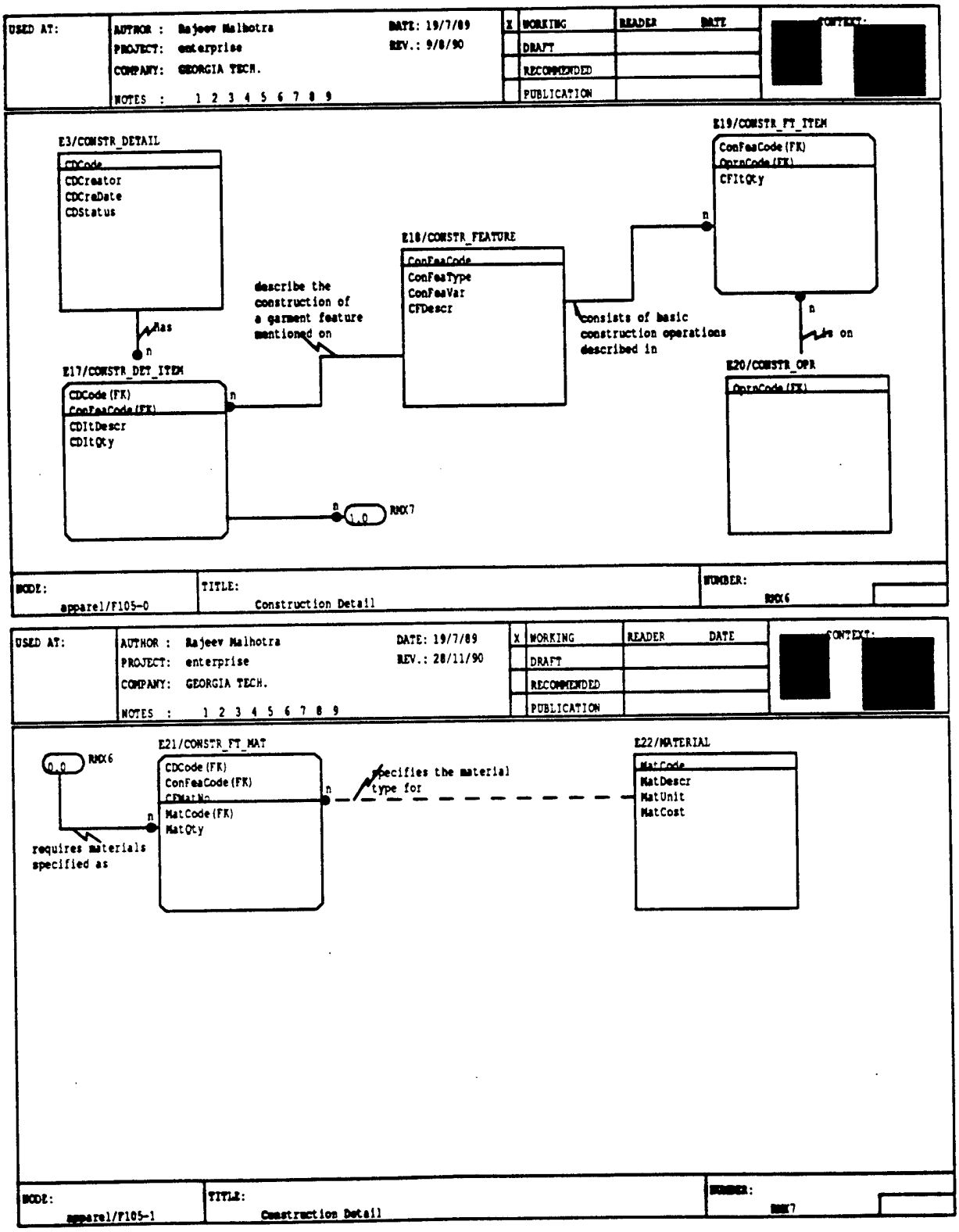


Figure 6.4. The Function View *Construction Detail*

ed and maintained by these services is used by other functions, such as product development, procurement and manufacturing. This information pertains to raw materials and their sources, manufacturing operations and resources, and quality standards and procedures. Data maintained on materials, which includes the description of the materials, their standard costs, the colors in which the materials are available and the sources for the materials, is modeled in the function view *Material Description* (Figure 6.2).

The resources of the enterprise and their capabilities are also modeled. For example, the data on equipment is modeled in the function view *Manufacturing Resources - Human* shown in Figure 6.5. The data on operators includes the operator's identification number, name, plant and department, skill level and capabilities. The entity *job* defines the skill level of operators. The operations that an operator has been trained to perform are modeled as the entity *operator skill*. Information maintained about each operation includes its description, standard time required to perform the operation, standard cost attributed to the operation and the skill level required to perform the operation.

6.2.3 Planning and Preparation for Production

Production planning activities include master planning for production and procurement of materials. Sales orders received from customers are central to the planning activity and are modeled as the entity *sales program* in the function view *Sales Program Description* (Figure 6.6). A sales program is instituted for the production of garments belonging to a particular style. In a sales program, the number of garments to be produced using each fabric is specified by the customer. The customer also provides the delivery schedule for the program. For example, a sales program may be instituted for the production of 10,000 trousers of a particular style - 2,000 of them in navy blue twill and the remaining 8,000 in khaki canvas. The customer may specify that 40% of the order be ready for delivery by November 1st and the remaining a month later. The sales program may not specify

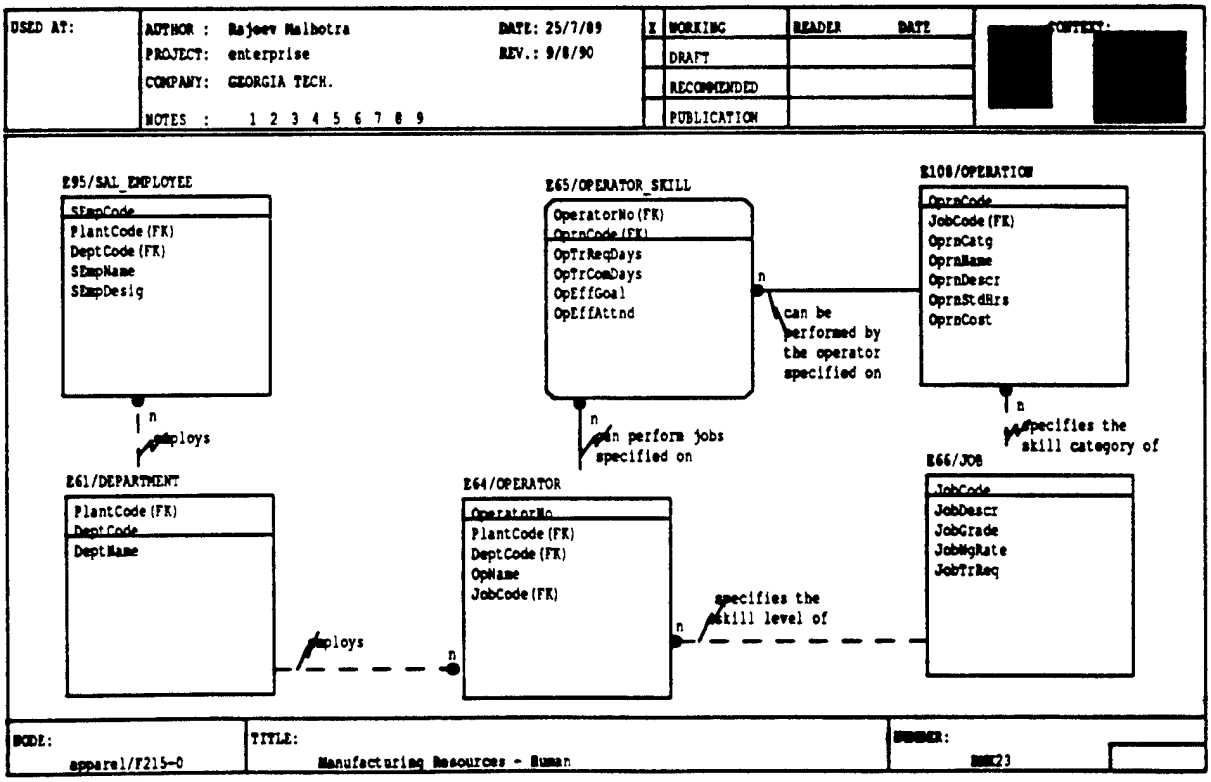


Figure 6.5. The Function View *Manufacturing Resources - Human*

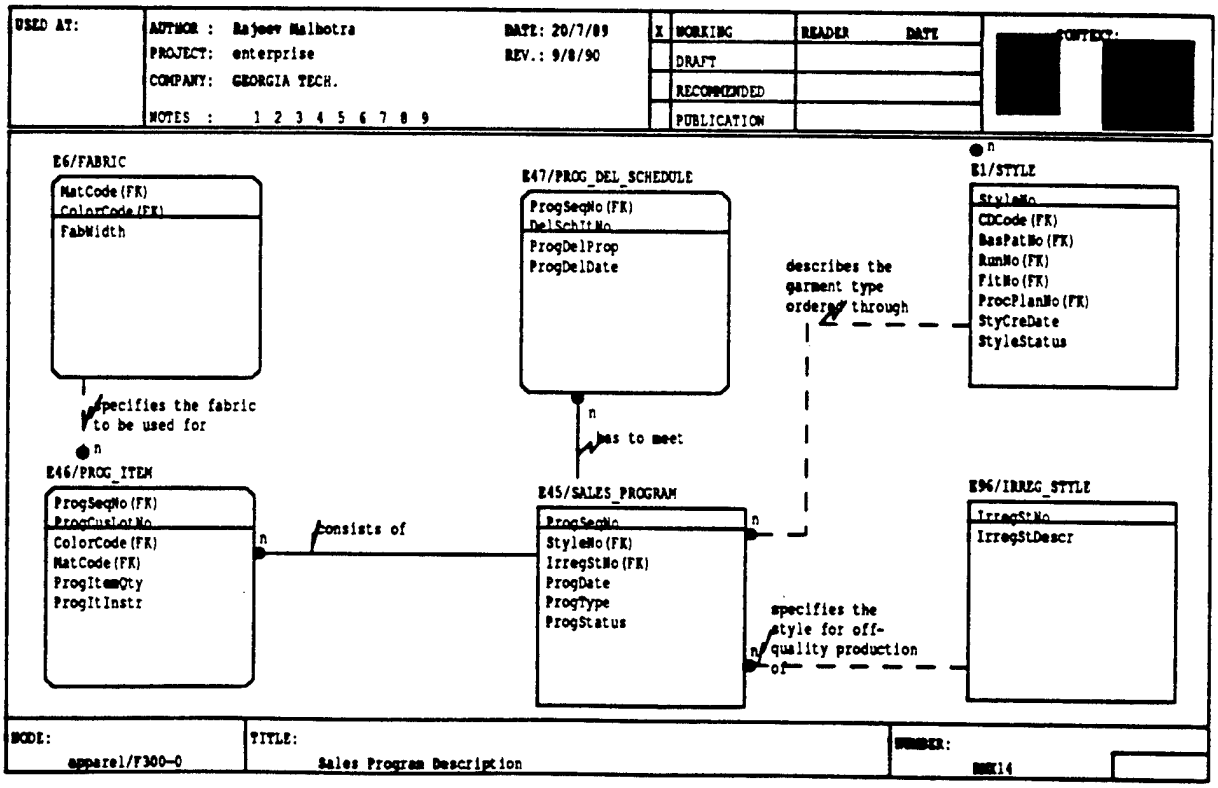


Figure 6.6. The Function View *Sales Program Description*

the size distribution of the order. The size distribution is not required for the master planning stage. Planning and preparation for production involves allocation of available production capacity to the program, assignment of colors for materials and ordering of materials for the program. On the master schedule (Figure 6.7), a part of the available production capacity is reserved for a program. The production plant is selected based on the available capacity and capability to produce a particular style. Production periods are selected to match the delivery schedule provided by the customer.

6.2.4 Production Control

Production control involves planning and release of production orders to manufacturing plants; the data maintained to support this activity is modeled in the function view *Cut Order Planning* shown in Figure 6.8. A production order is issued by production control to manufacturing for producing the specified quantity of garments in a given color and size distribution towards the completion of a sales program. Production for a sales program is completed through one or more production orders. A marker, consisting of an arrangement of sections, each of which is packed with graded pattern parts for one or more sizes, is assigned to a production order. As part of cut order planning, the spread layout is also created. The spread layout specifies the number of layers of each fabric type/color that should be spread under each marker section to yield the desired number of garments in each size and color, and is modeled as the entity *spread section*.

Production orders are prepared and released after taking into account the finished goods inventory, inputs from customer regarding size and color distribution, and availability of raw materials and production resources. Each manufacturing plant has a schedule on which the orders released for production at that plant are posted. Such a schedule may be prepared and released on a weekly or biweekly basis. Each manufacturing plant receives its production schedule whereas a centralized cutting facility receives the schedules for all the

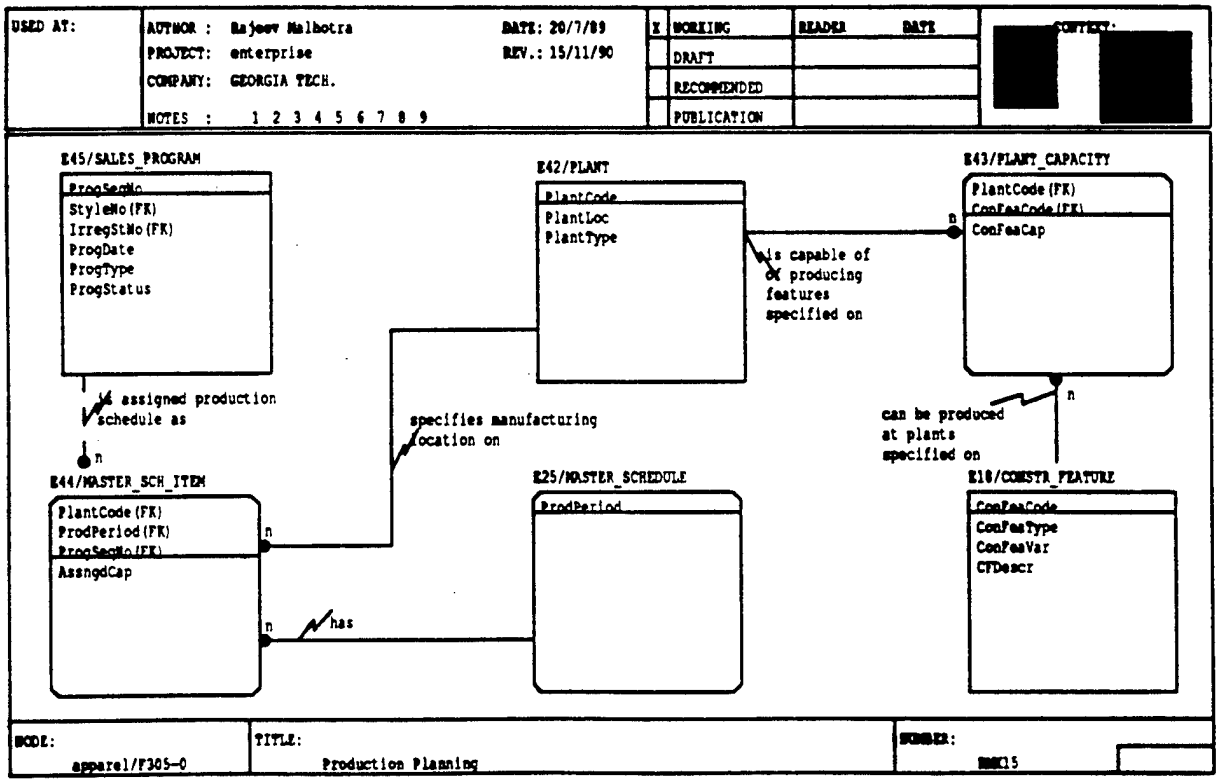


Figure 6.7. The Function View *Production Planning*

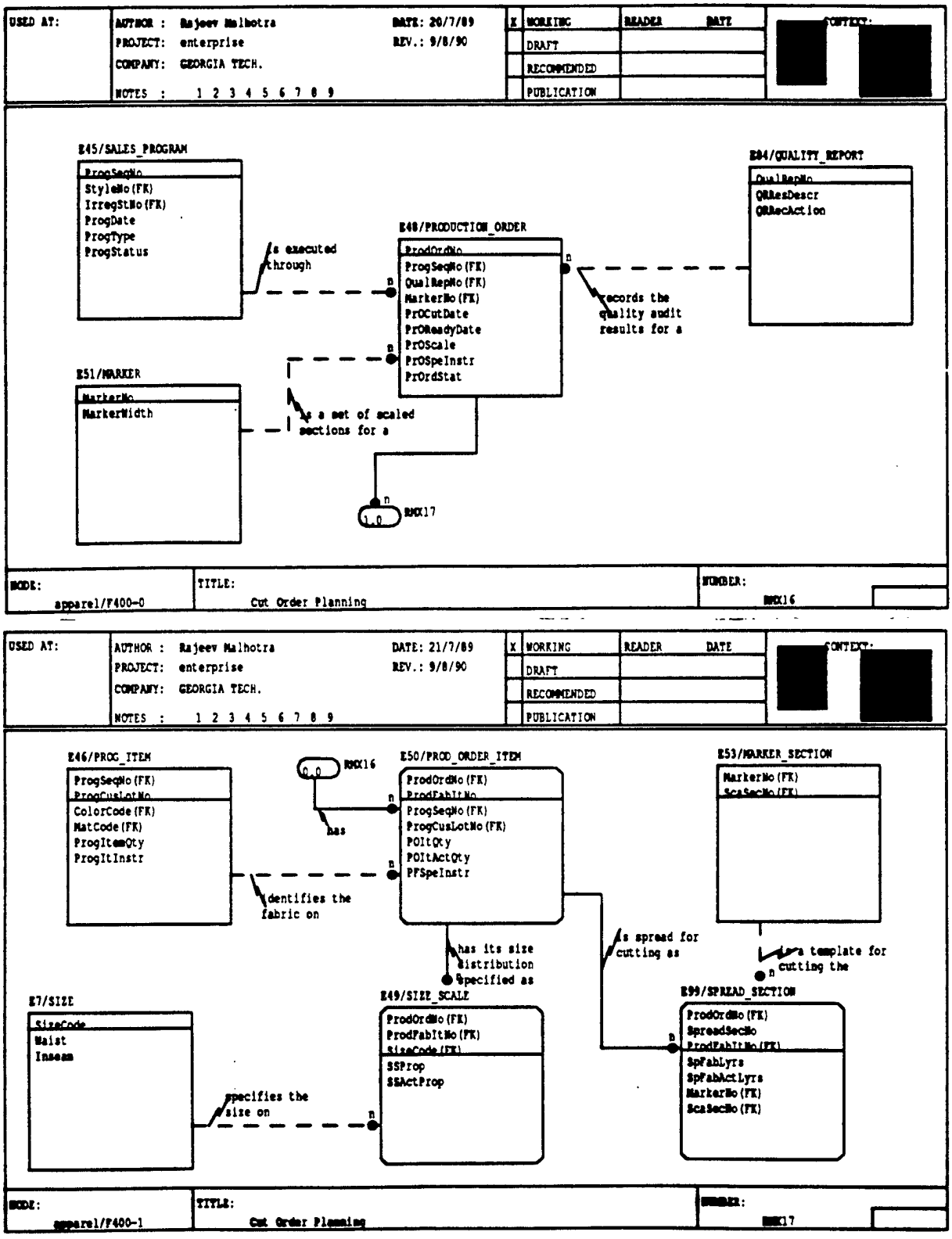


Figure 6.8. The Function View *Cut Order Planning*

plants it serves.

6.2.5 Manufacturing

Manufacturing of garments involves cutting, sewing and finishing. Cutting for production orders is scheduled in such a way that the production schedule prepared by the production control function can be followed. The cutting facility collects all the fabric parts and other materials required to produce garments for a production order and ships this package to the appropriate plant for further processing. The fabric parts are labeled after cutting so that the parts belonging to any one garment can be easily identified (Figure 6.9).

In the manufacturing plants, each scheduled production order is assigned groups of equipment on which the garments are assembled and finished (Figure 6.10). A group may be a line or a module. Each group is assigned a set of operations and operators. For example, four modules may be assigned for a production order, one each for trouser front assembly, back assembly, final assembly and finishing. The operations assigned to each equipment group are specified in terms of process steps on the *process plan* for the style.

6.2.6 Distribution

Distribution of finished garments to customers involves stocking of goods received from the plants, retrieval of appropriate garments for shipping, and packing of these goods. The entities supporting stocking of finished goods are modeled in the function view *Finished Goods Warehousing* shown in Figure 6.11. Regular quality finished garments are stored in small containers each of which contains one or two dozen garments of the same color and size. Irregular garments are collected together in large bags and are not stored with regular garments in the warehouse, but are moved to a special area in the distribution center for disposal. Containers with regular quality garments are divided into groups for storage since a single storage location may not be large enough to hold all the containers for an order. Each location may hold groups from many production orders.

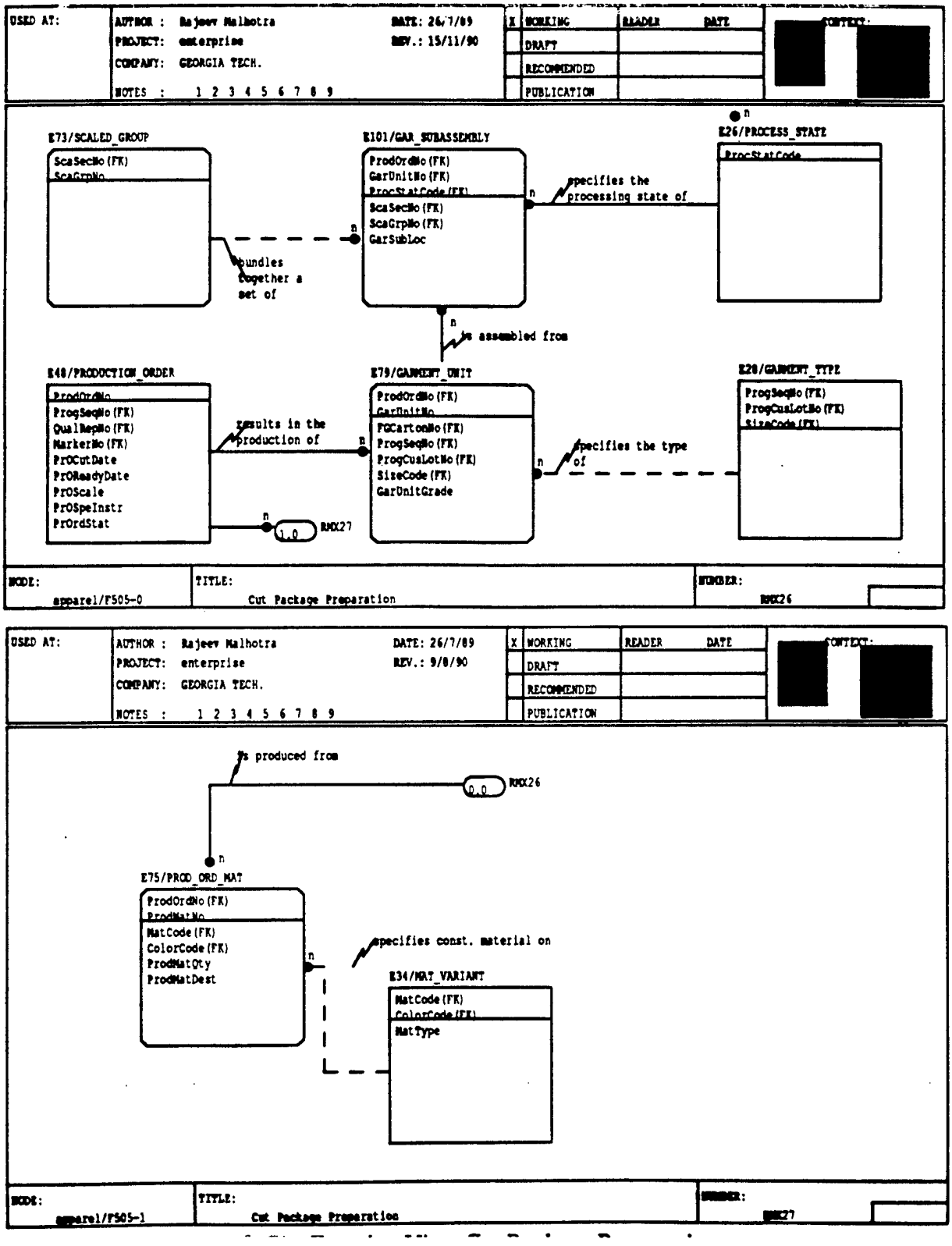


Figure 6.9. The Function View *Cut Package Preparation*

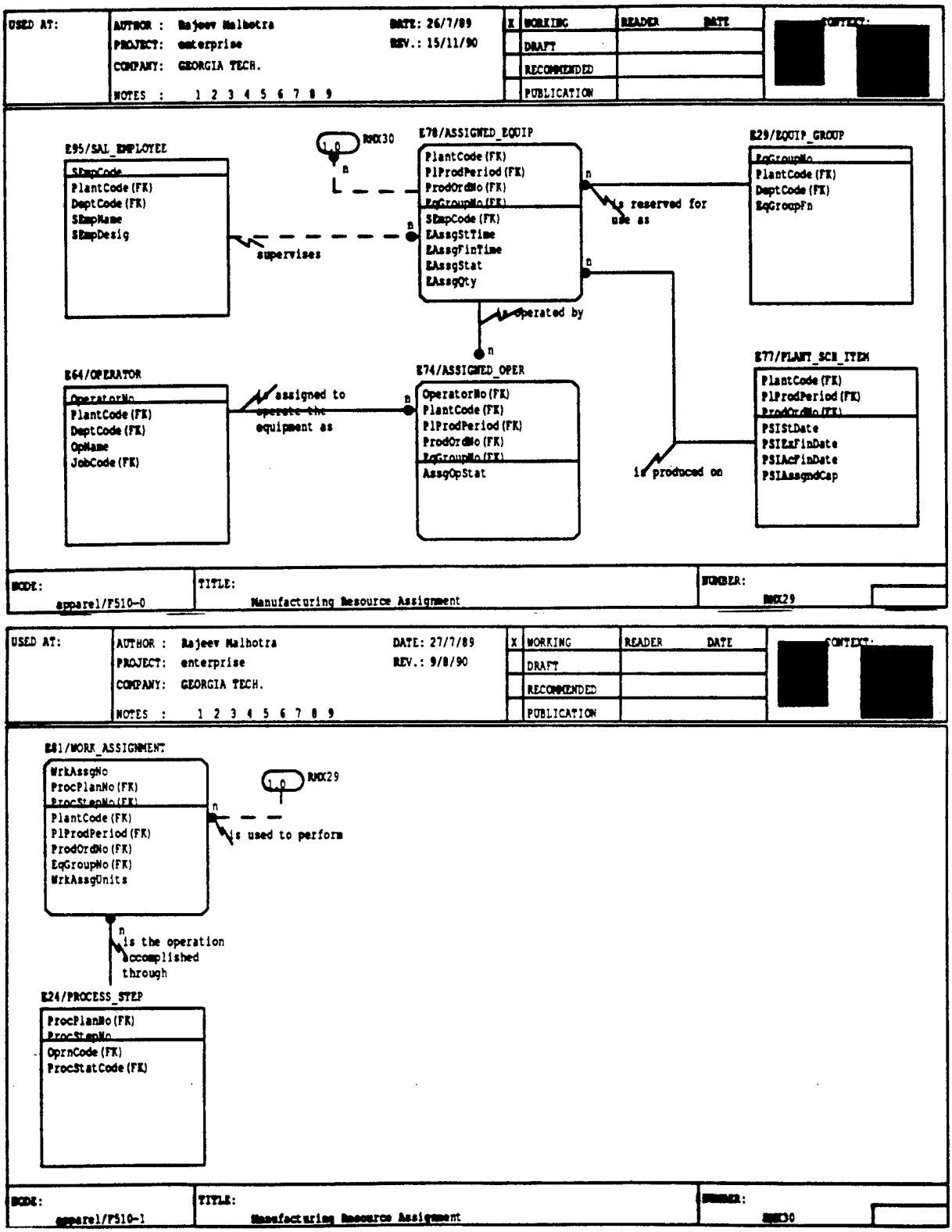


Figure 6.10. The Function View *Manufacturing Resource Assignment*

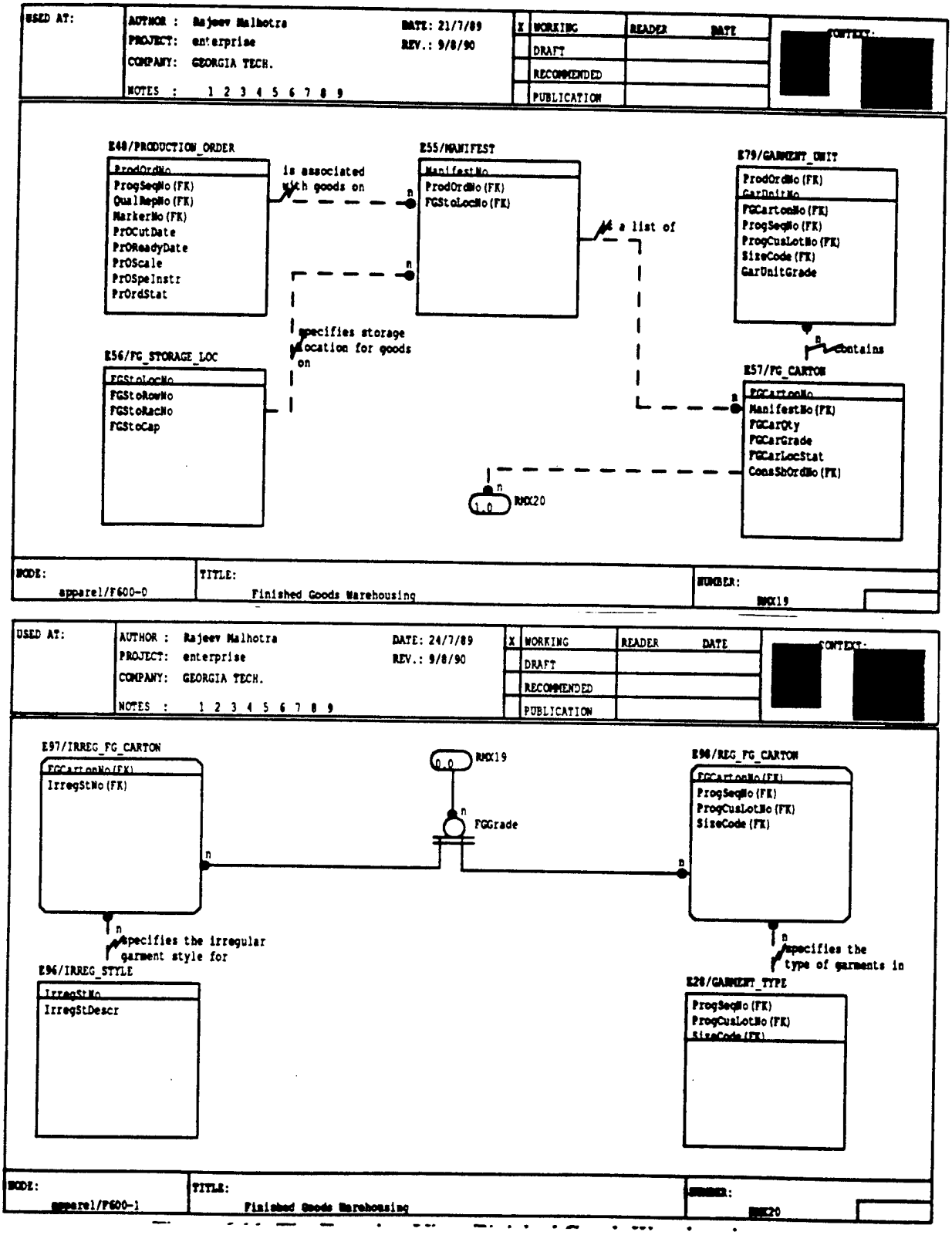


Figure 6.11. The Function View *Finished Goods Warehousing*

Garments produced for a sales program are shipped to the customer only when shipping orders are received from the customer. A shipping order specifies the mix of colors and sizes of the ordered style (Figure 6.12). Typically, a customer sends a shipping order for each retail outlet. Garments for each shipping order are packed individually in shipping cartons and shipped either directly to the retail outlets or to distribution points specified by the customer. To facilitate packing and shipping, all the shipping orders for a sales program that have to be shipped around the same time are consolidated into a single packing order and garments required for these orders are retrieved together.

6.3 Role of TO BE Information Model in CIM Architecture

The interactions between the various functions of an enterprise involve exchange of information between them which can be accomplished through one of the following three means: paper, electronic data files and information sharing. The traditional method is to use paper for exchanging information. For example, sketches and engineering drawings are used to communicate the design of a product to be manufactured by the design department. Paper-based transfer of information is not only slow but also is error-prone. When updates are frequently made to the documents, it becomes difficult to ascertain that all the intended recipients have the same updated version. Transfer of information using paper was the primary means of formal communication when computer-based systems were not available to support the enterprise functions. However, this method of information transfer is still very commonly used in organizations where computer-based systems abound, because protocols for exchanging information between computer systems may not exist. For example, a design produced on a sophisticated CAD system, may still have to be sent to process planning and manufacturing as a paper printout.

The second method for exchanging information between functions is as electronic data. Means for transferring data files between computers are available and widely used.

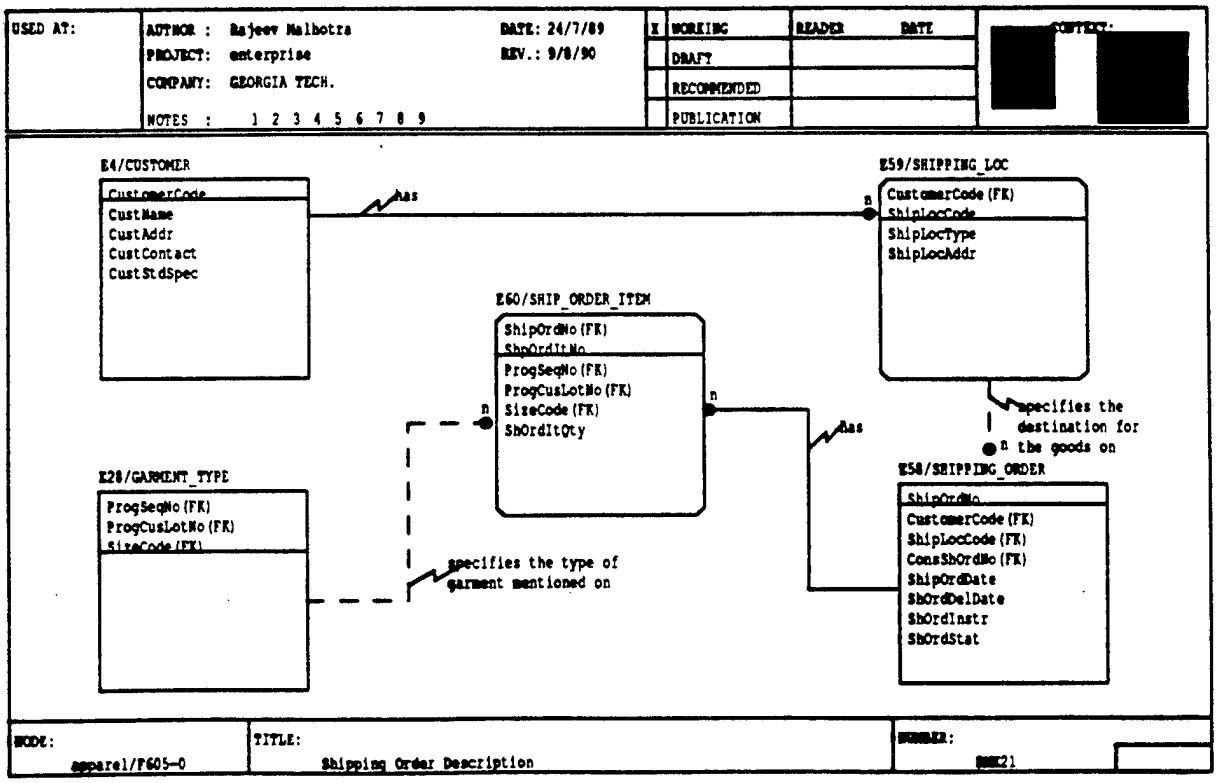


Figure 6.12. The Function View *Shipping Order Description*

Unlike paper transfer, electronic data interchange is quick. However, just the ability to interchange data between computers does not solve all problems associated with information transfer. Electronic data interchange is meaningful only if data from one function can be interpreted by other functions. For example, a process planning system can use an electronically transferred design data file from a CAD system only if it can interpret it. Even if the means for interpreting data exist, electronic data interchange between the design and process planning functions in the current example would result in creation of two copies of the design data - the CAD copy and the process planning copy. Since data interchange is a batch process, inconsistencies between the two copies is likely to exist if the data in one of the copies is updated. Thus, even with electronic data interchange, the problem of multiple, inconsistent copies of the same information is not eliminated.

The third method of exchanging information is through sharing of data between inter-linked functions. This method is the foundation of computer-integrated manufacturing. With this method, information to be exchanged is maintained in a single data set from where it is accessed by all the functions. For example, the design information is maintained in one data set which is updated by the CAD system and seen by the process planning and other systems requiring access to design information. Thus, with the data sharing method, the cause of inconsistent data is eliminated. However, data sharing cannot work if the data residing in a single data set is defined from the point of view of a single system, e.g., CAD. It is necessary that the data be viewed at the enterprise level instead of the function level so that data definitions that are compatible with the needs of all the functions sharing information could be developed. The TO BE information model provides such a view of the data maintained to support the functions of an apparel manufacturing enterprise. An example from the TO BE information model illustrates how the model supports information sharing.

Information about materials, such as fabric, trim, accessories, etc., from which garments are manufactured, is processed by numerous functions in an apparel enterprise.

Different types of materials are defined in the TO BE information model (Figure 6.2) as category entities of the entity E22/MATERIAL which, along with its dependent children entities, defines the structure of the single data set in which information on materials is maintained. Materials are defined based on specifications created by the product development department to meet the styling requirements of the customers. The entity E22/MATERIAL is used in the definition of the construction detail (Figure 6.4) where it specifies the material used for constructing a feature of a garment. Another function that accesses information about materials is the vendor development function which selects suitable suppliers for each material (Figure 6.2). The information about selected vendors for each material is structured in the entity E94/MATERIAL_SOURCE which is modeled as a dependent child of the entity E22/MATERIAL. Variations on a material, e.g., different colors for a type of fabric are defined by the entity E34/MAT_VARIANT, which is modeled as a dependent child of E22/MATERIAL. E34/MAT_VARIANT is used to specify the construction materials, such as trim, buttons, thread, etc. whose color is dependent on the fabric color (Figure 6.13). The material procurement function uses E34/MAT_VARIANT to specify the materials to be procured on a purchase order (Figure 6.14). The same entity is also used to specify the materials to be shipped to manufacturing plants along with cut fabric parts for producing garments (Figure 6.9). Thus, in the TO BE information model, the entity E22/MATERIAL and its dependent children entities E34/MAT_VARIANT and E94/MATERIAL_SOURCE provide the only definition of the data on materials in the entire model.

As illustrated by the current example, the TO BE information model provides a single integrated definition for the apparel enterprise data. This definition is the conceptual schema for an enterprise-wide information system which can support information sharing.

6.4 IDEF$_1$x as a High-Level Data Definition Language

The TO BE information model entities define the relational database tables (rela-

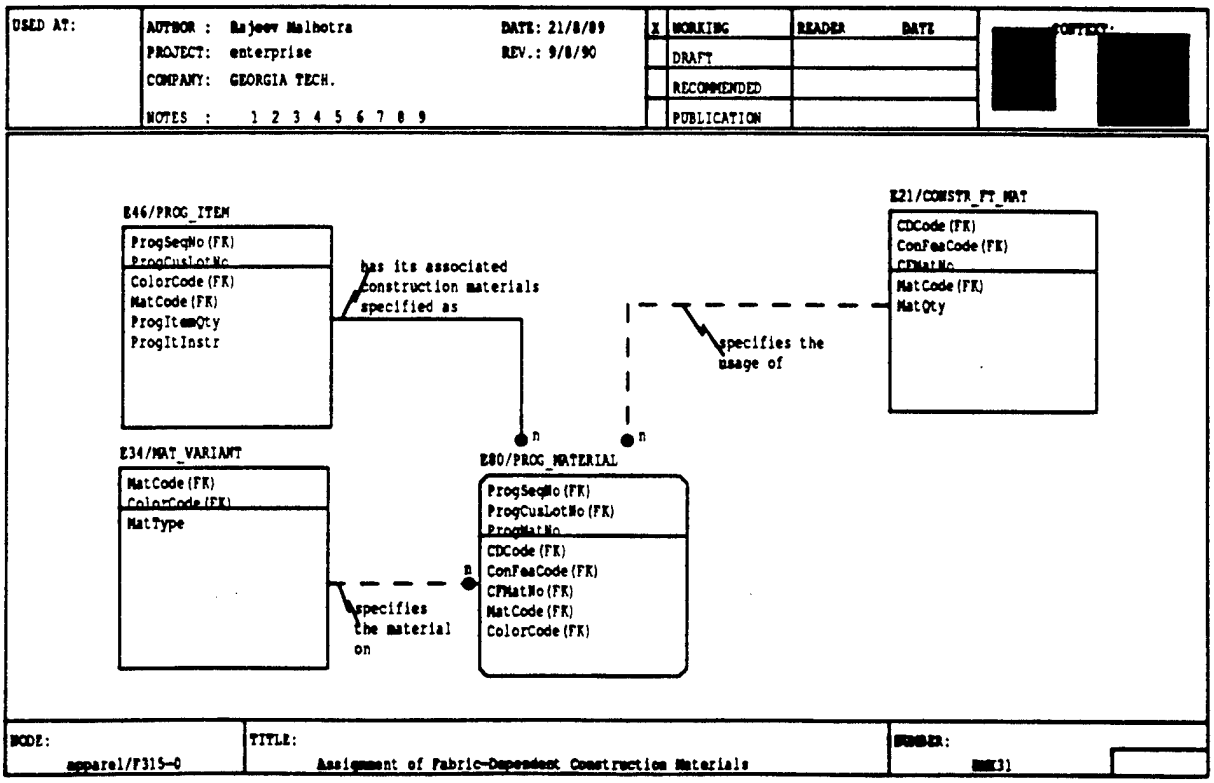


Figure 6.13. The Function View *Assignment of Fabric-Dependent Construction Materials*

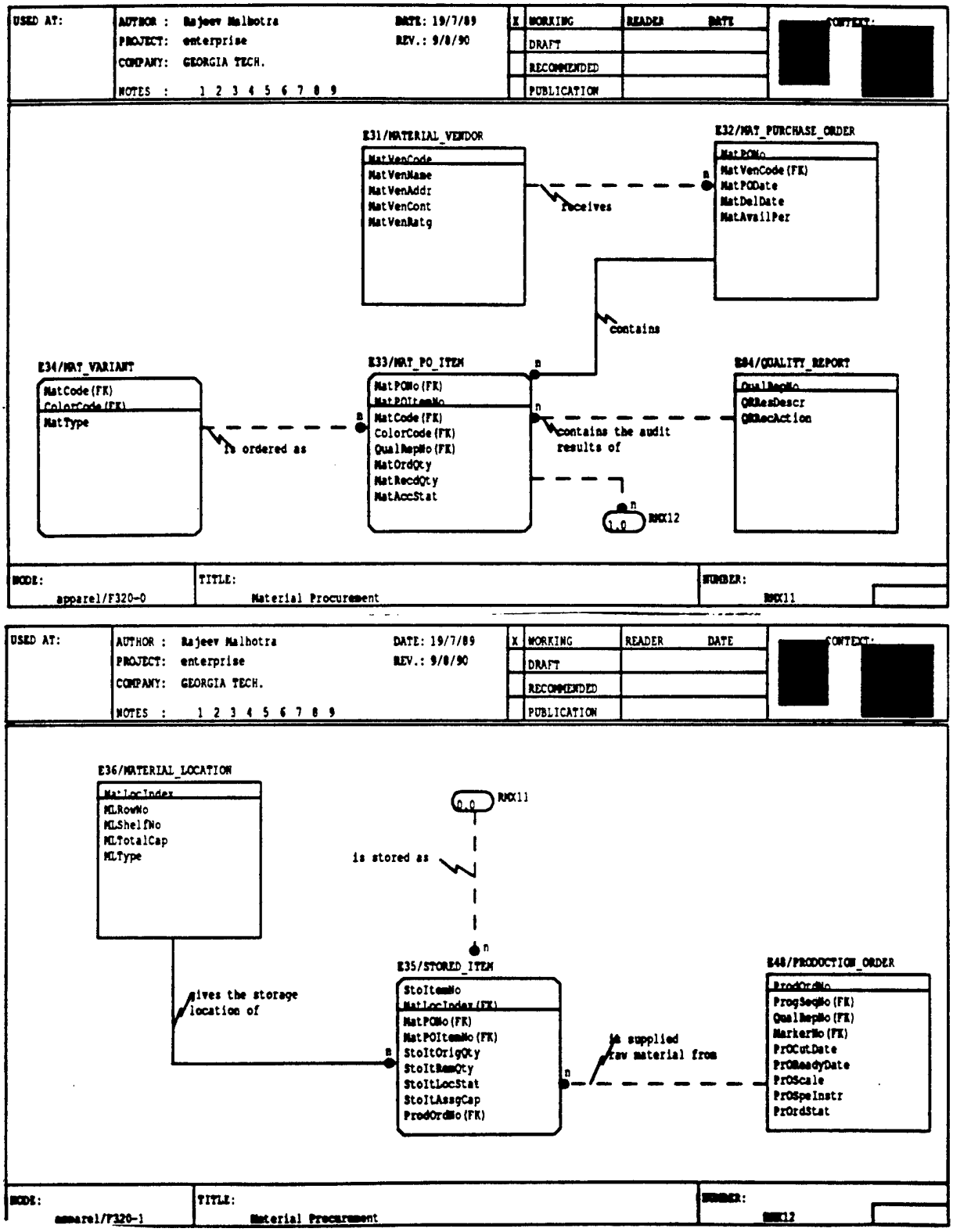


Figure 6.14. The Function View *Material Procurement*

tions) in which the enterprise data could be maintained. The attributes within the entities correspond with the data fields within each table. Tables in relational databases are usually set up using SQL, a declarative data definition and manipulation language. A software utility called SQLGEN was developed to control the relational database schema creation and maintenance process through an IDEF$_1$x model. The modeler defines the conceptual schema as an IDEF$_1$x model using the IDEFine-1 tool. SQLGEN converts the IDEF$_1$x model into SQL commands for setting up or modifying the relational database schema.

6.4.1 SQLGEN Features

The SQLGEN utility performs the following functions:

1. It scans the IDEFine-1 information model data files to generate SQL data definition commands for setting up a relational database corresponding to the model.
2. It provides debugging and diagnosis features for checking the integrity of the information model that are not available in the IDEFine-1 software.
3. It provides the means for maintaining the database. The database that is set up initially gets out-dated when changes are made to the information model. SQLGEN assists the database administrator in modifying the database to bring it in conformance with the information model.
4. The utility also generates a concise listing of the entities, attributes and their definition in a table form. This listing can serve as the data dictionary for the model.

6.4.2 SQLGEN Design Approach

A modular approach was followed in the design of the interface software. The complete utility consists of 6 modules each of which performs a specific function and generates data for the subsequent module. The utility is executed through a DOS batch file script which provides an easy-to-use interface to the utility.

The first module called SQLGEN1 scans the data files of the IDEFine-1 model

and generates a table listing entities, their attributes and the attribute definitions. This table is an ASCII file which is formatted for hard copy printout. The second module called SQLGEN2 reads the condensed model data and scans it for missing or incorrect attribute definition. Since IDEFine-1 does not do the consistency checking for attribute definition, this task is also done by SQLGEN2 module which takes the attribute definition from the entity where that attributes is owned and uses that definition consistently everywhere else. Also, the IDEFine-1 attribute types are replaced with the SQL (ORACLE) types. After checking the model, this module generates a map containing all the attributes that pass the checking. It also generates a log file containing messages regarding incorrect or missing attribute definitions. The log file pinpoints exactly where the correction needs to be made in the model. The map generated by SQLGEN2 is used by the third module SQLGEN3 for generating the SQL commands for setting up the database tables. The attributes that form the primary key of an entity are converted into columns in which null values are not allowed. A table corresponding to an entity is set up in such a way that it may be referred to by the entity name or entity number. The generated commands are stored in a file which can be executed by an SQL interpreter to set up the database. Although, the utility has been tested on ORACLE database system, it may be used with any other DBMS which uses ANSI standard SQL as its data definition language (DDL).

The remaining modules implement the update features. Module SQLGEN4 controls the generation of the model map. The first three modules are executed only if the information model has been modified since the model map was last generated. The next module SQLALT1 compiles the information regarding the changes made to the information model since the last time the utility was run, and passes it on to the last module SQLALT2. SQLALT2 looks at this information and applies a set of rules to it to generate the updates. The SQL commands for alterations such as changes in entity names, addition of new entities and attributes, and addition of attributes to existing entities are automatically written to

an output file. There are other alterations such as change of attribute name or type, and deletion of attributes which are data dependent. Appropriate messages regarding such changes are written to the alteration log which can be used by the DBA to modify the database.

CHAPTER VII

TO-BE ARCHITECTURE: THE FUNCTION MODEL

The concept of sharing non-redundant data does not necessarily require that the data be maintained in a central database. It only requires that the information systems of all the functions conform to a single conceptual schema and be able to share the data that they generate by exporting it to other functions. This approach is adopted by distributed information systems and is better suited for large enterprises than the central database approach [Groover87]. The conceptual schema provided by the information model is an essential part of the specification for a distributed information system. In addition, there is a need to understand where, in the functional structure of the enterprise, the data defined in the information model is accessed and specify how the various enterprise functions interface with the distributed information system. Such interfaces for the functions of the TO BE apparel manufacturing enterprise were modeled in the TO BE function model presented in this chapter.

7.1 Model Syntax and Conventions

The TO BE function model was developed using the IFEM methodology discussed in Chapter V. In the IFEM methodology, the function model is developed using $IDEF_0$'s cell modeling technique discussed in Chapter IV. In addition to modeling the function structure, the ICOM interfaces between the functions are defined using the data definitions from the TO BE information model. By sharing the entity definitions between the information and function models, the two models are integrated into a single architecture

of the static structure of the proposed CIM system for an apparel enterprise. The need for aggregating the IDEF$_1$x entities defined in the TO BE information model into object classes was discussed in Chapter V. The ICOMs are defined in terms of the object classes developed as an extension of the information model. The syntax used to model the object classes is discussed in this section.

7.1.1 Syntax of Object Class Definitions

Object Classes: An object class corresponding to each IDEF1x entity in the TO BE information model is defined (Appendix B). Object classes are identified by the letter 'F' and carry the same identification number and name as their equivalent IDEF$_1$x entity. For example, object class *style* is identified as F1/STYLE. Each class is defined as a collection of features.

Features: The features of a class are identified by short descriptive names that appear to the left of colons in the object class definitions. For example, features of object class F1/STYLE are *ID*, *Date*, *Status*, *CD*, *Fit*, *Pattern* and *ProcPlan* (Figure 7.1). The definitions of the features appear on the right-hand side of the colons. Features representing atomic data items are defined in terms of their equivalent attributes in the entity definitions. For example, the feature *Status* in F1/STYLE, which is same as the attribute *StyleStatus* in E1/STYLE is defined as:

Status : E1.StyleStatus;

Relationship between objects classes are expressed by defining a feature of one class as an object of another class. For example, the relationship between the entities STYLE and PATTERN is expressed by defining a feature *Pattern* in class STYLE to be an object of class PATTERN as follows:

Pattern :F14{PATTERN};

Multi-valued features are defined as a list of objects in a particular class. For example, in

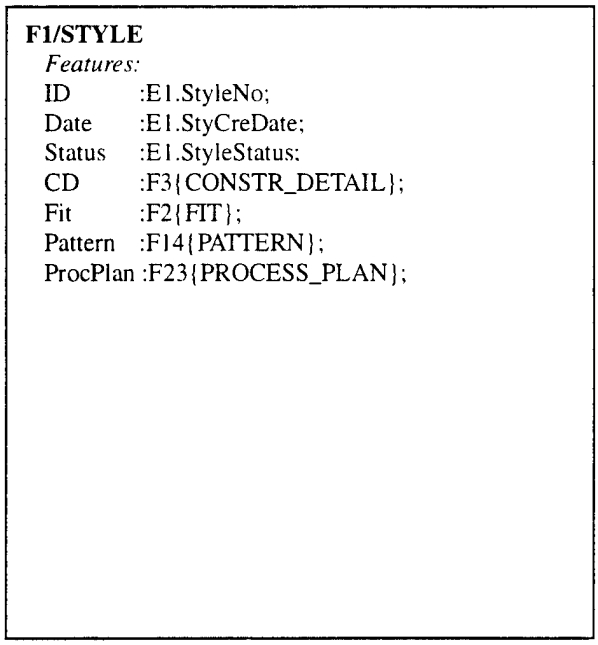


Figure 7.1. Definition of Object Class *STYLE*

the class F14/PATTERN, the feature *Part*, which represents a collection of pattern parts is defined as follows:

```
Part                    :LIST(F15{PATTERN_PART});
```

Derived features are defined in terms of type of value returned by the feature. The procedure for deriving the value is not specified as it is implementation-specific. Comments are used to indicate what the feature returns. For example, the feature *Cost*, which returns the cost of construction for a garment derived from the costs of materials and operations used in garment construction, is expressed as follows:

```
Cost    :(The sum of costs of materials and operations on the construction detail)
```

Constraints: Constraints on features are expressed using the WHERE clause in the definition of a feature. A relational expression defining the constraint on the feature follows the WHERE clause. For example, if only a validated pattern could be assigned to a style, this constraint could be expressed as follows in the definition of the feature *Pattern* in the class F1/STYLE:

```
Pattern     : F14{PATTERN}
              WHERE F14{PATTERN}.Status IS 'VALIDATED';
```

Inverse Relationships: Constraints are also used to express inverse relationships between classes. For example, for the entity F15/PATTERN_PART, the feature *Pattern* identifies the pattern to which the part belongs. This relationship is inverse of the relationship between pattern and pattern part expressed by the feature *Part* in the entity F14/PATTERN. The inverse relationship in this case can be expressed as follows:

```
Pattern     : F14{PATTERN}
              WHERE SELF IN F14{PATTERN}.Part;
```

Keyword SELF identifies the object class in which the feature is defined.

7.1.2 Syntax for ICOM Definitions

Structure Definitions of ICOMs: The structure definitions of ICOMs modeled in the TO BE function model are contained in their respective glossary entries (Appendix C). For example, the ICOM *Production Schedule* is defined as (Figure 7.2):

F76{PROD_SCHEDULE};

Constraints on objects in ICOM definitions are expressed in an easily understandable syntax using the WHERE clause. For example, the ICOM *Style - Validated*, which represents a style that has been validated, is defined as follows:

F1{STYLE} WHERE Status IS 'VALIDATED';

A feature of an object is referenced in an ICOM definition by prefixing it with the name of the object class to which it belongs and separating the two names with a period. For example, the ICOM *Style's Process Plan*, which represents a process plan assigned to a style is defined as:

F1{STYLE}.ProcPlan;

An ICOMs that represents a set of features is defined as a partition that limits the view of the ICOM to specific features in a class. For example, the ICOM *Style's Shape*, which represents the features *Pattern* and *Fit* of the class F1/STYLE, is defined as follows:

(Pattern, Fit) FROM F1/STYLE;

Type classification of ICOMs: Defining ICOMs in terms of database structures implies that the functions represented in the function model interface with each other only through the database. However, in an IDEF0 function model, there may be ICOMs that represent direct exchange of transient information between functions. For example, ICOMs may represent start, stop and acknowledge signals that are sent from one process to another as part of a hand-shaking protocol. Data on such transient entities need not be maintained as it ceases to be of interest once it has been used by the receiving function. Consequently, transient entities are not included in the information model. In IFEM, ICOMs are classified

Production Schedule
Type: S/P
Structure:
F76{PRODUCTION_SCHEDULE};
Description:
Schedule for production (assembly and finishing) plants for a particular production period.

Figure 7.2. Glossary Entry for the ICOM Interface *Production Schedule*

as *transient* or *persistent*, based on the type of entities they represent. The data structures that define the transient ICOMs are not related to the information model entities. In the TO BE function model glossary, transient ICOMs are identified by letter 'T' and persistent ones by letter 'P' under the heading *Type* in the glossary entries for the ICOMs. For example, the type code of *Production Schedule* contains 'P' indicating that the ICOM represents a persistent entity (Figure 7.2).

Another classification of the ICOMs results from the fact that ICOMs may represent abstract ideas or knowledge that cannot be defined as *structured* data in the information model. For example, the knowledge about fashion trends constrains the garment design function, but is difficult to represent this knowledge as structured data. ICOMs representing unstructured entities are classified as *free-form* ICOMs and are not assigned any data structures in IFEM. The type code for free-form ICOMs is 'F' and for structured ICOMs is 'S'. In Figure 7.2, 'S' in the type code of *Production Schedule* indicates that the ICOM represents a structured entity.

7.2 TO BE Function Model

Appendix C contains the TO BE function model representing the function structure of a computer-integrated apparel manufacturing enterprise. The activities of the apparel enterprise were analyzed through the AS IS model and presented in the TO BE function model as functional components of an enterprise-wide information system that create, modify and reference data. Functions that transform physical entities, e.g., *Cut Fabric*, have been abstracted as information processing functions that modify the data representing the state of the physical entities. The TO BE model adheres to the context and viewpoint of the AS IS model. The methodology outlined in the previous section was used to integrate the TO BE function model with the TO BE information model discussed in Chapter VI. In the resulting integrated model, the ICOM interfaces, in addition to representing inter-con-

nections between the functions, also represent the interface of the functions to the enterprise-wide CIM information system. The model complements the TO BE information model in defining an architecture for a distributed CIM information system in that it identifies the enterprise functions that require access to each data entity defined in the TO BE information model.

The TO BE function model incorporates refinements to the AS IS model to facilitate a distributed information sharing architecture for CIM. The modifications, based on the analysis of the AS IS function model and the TO BE information model, have resulted in a reorganization of the function structure of the enterprise, elimination of redundant functions and generalization of the model for apparel manufacturing. These modifications are discussed in this section with the help of illustrative examples from the model.

7.2.1 Restructuring of the Function Hierarchy

In the TO BE function model, the AS IS model function hierarchy was restructured to group the enterprise functions around the data entities they access and modify. In the restructured hierarchy, the outputs of a function represent the data entities that are made available by that function to the outside functions. The ICOMs that interface the sub-functions of a function represent data that is local to that function and is not seen by the outside functions. The local and shared data are distinguished at all levels in the TO BE function hierarchy.

In the top level diagram A0 (Figure 7.3), the six functions, A1 through A6 correspond to the six functional areas into which the information model is divided. Each of these functions creates and modifies data entities modeled in the function views in its respective functional area. From the viewpoint of a function, there are two types of data entities: shared and local. Only those data entities that are of use to other functions are shared by a function; the rest are viewed as local data entities that are unavailable to the outside func-

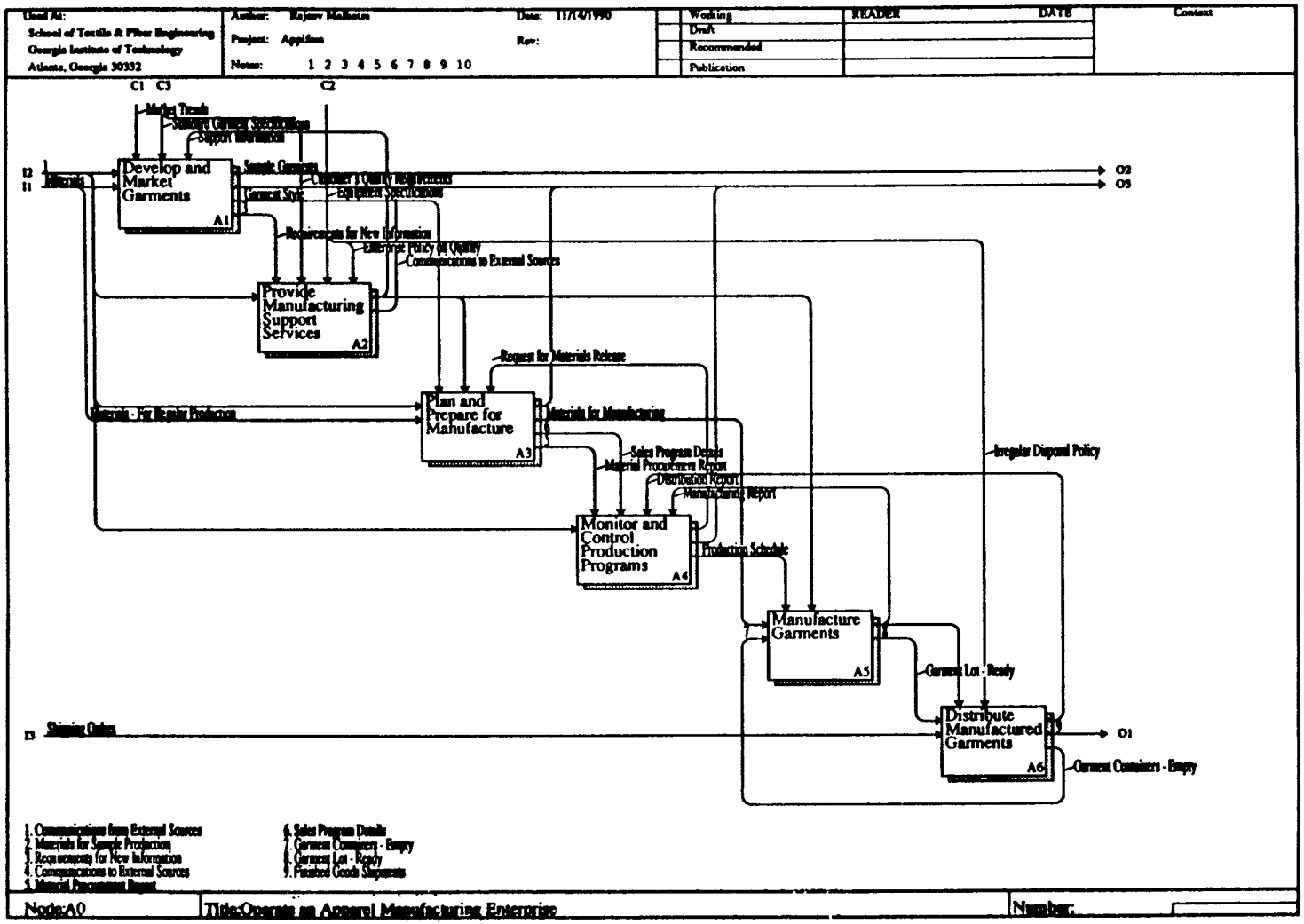


Figure 7.3. The top-level diagram of the TO BE function model

tions. For example, the function *Develop and Market Garments* (A1) in Figure 7.4 is viewed as a function that creates and modifies a garment style based on customer's requirements. An outcome of this activity is a garment style which is shared with other functions. A style has components, such as construction detail, pattern, fit and process plan, which are created and modified by the sub-functions of A1. Data on all these components is maintained locally within A1 and is not available outside it. Only those patterns, construction details, fit and process plans that have been assigned to the styles represented by the output of A1 are shared with the outside functions. The output interface *Garment Style* of function A1 is defined in the model as a style that has been validated. This constraint in the definition of garment style applies to the shared information as well. Therefore only the validated styles are available outside A1.

The distinction between local and shared data is made for functions at all levels in the model hierarchy. For example, consider the sub-functions of A1 in Figure 7.4. The data on style concepts generated by the stylist is local to the function *Market Style Ideas to Customers* (A11). Only those style concepts that have been selected by the customer for development into styles are shared by this function with *Develop Garment Style* (A12). Similarly, data on sample production schedules that is of relevance only to the sample production functions, is local to the function *Provide Sample Garments* (A14).

The output arrows represent the data entities exported by a function while the input and control interface arrows represent the data entities imported by the function. The imported data entities are either referenced or used to generate the outputs of a function. The data accessible to a function is constrained by the definition of the input and control interfaces. For example, consider the function *Receive and Confirm Sales Order* (A31) which has *Style - Validated* as one of its control interfaces (Figure 7.5). The function requires access to the details of the styles that are to be assigned to the new sales programs. As per the definition of the ICOM *Style - Validated*, function A31 has access to all the data

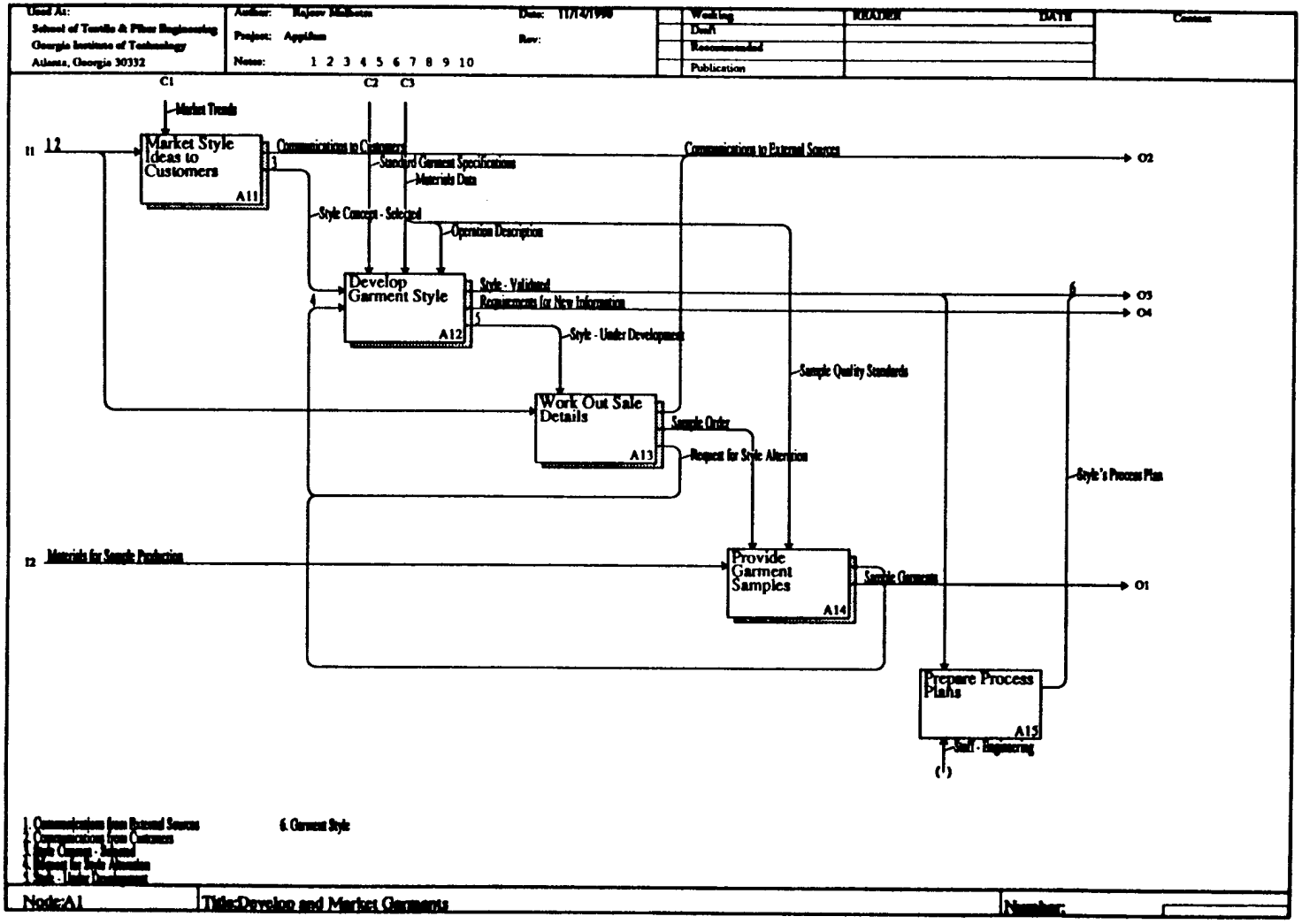


Figure 7.4. Details of the *Develop and Market Garment* function

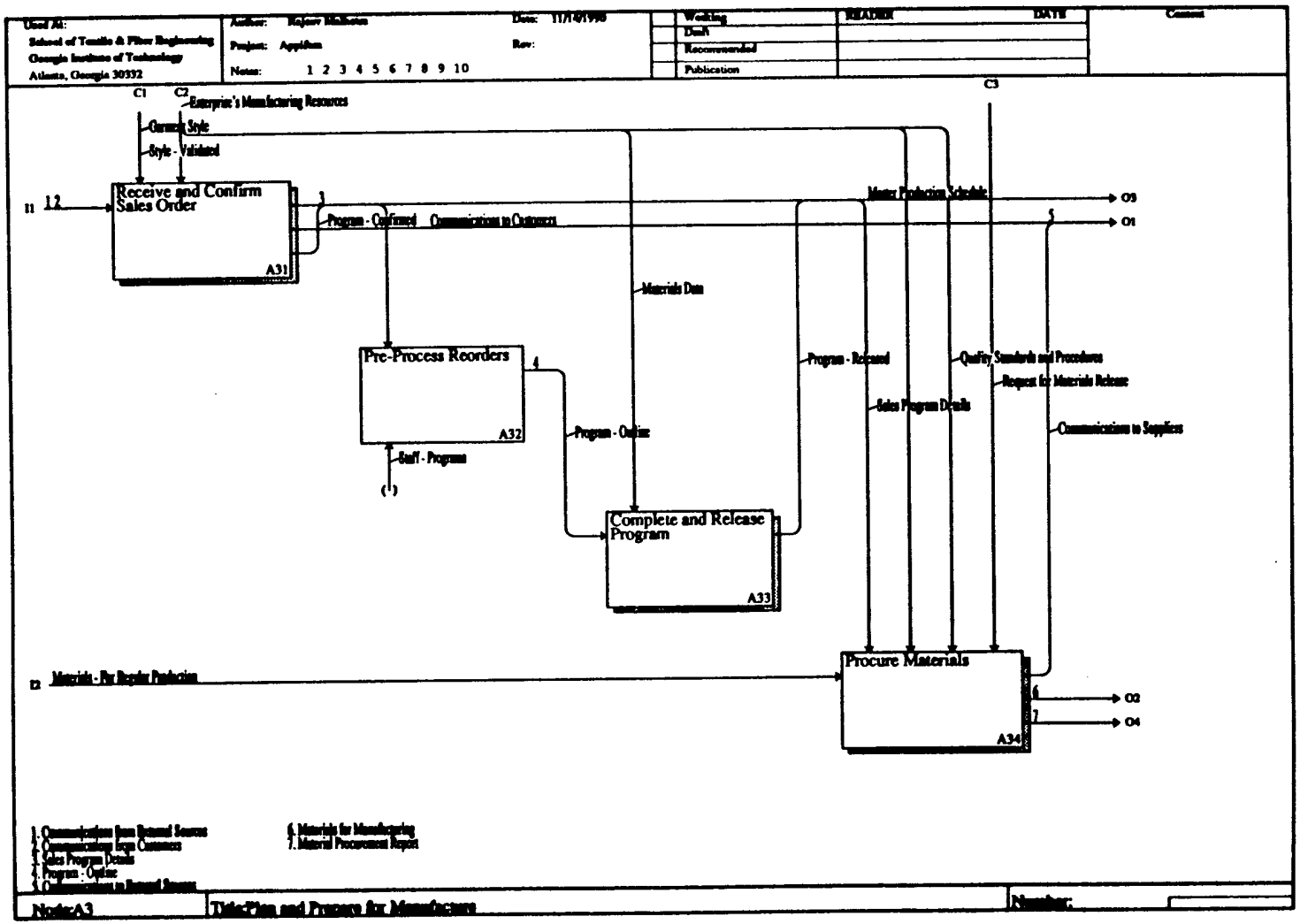


Figure 7.5 Details of the *Plan and Prepare for Manufacture* function

on styles that have been validated. The accessible data entities recursively include the components of the entity style, e.g., data on style would include all the data on the fit of the style and all the data that describes the grade table feature of this fit.

Thus, using the ICOMs, the interface of the functional components of a CIM information system for an apparel manufacturing enterprise were specified in the TO BE function model.

7.2.2 Elimination of Redundant Functions

In the AS IS enterprise, due to the absence of an integrated information system, activities for handling the paperwork and checking the validity of available information are part of the enterprise functions. Such activities were represented in the AS IS model, but are not included in the TO BE model. For example, consider the function *Validate Model* (A223) in diagram A22 of the AS IS model in Figure 7.6. This function ensures that style data used to complete the details of a program is consistent with the style data maintained by the product development function. This function is not included in the TO BE model since the validation of the styles is performed automatically during the product development stage. A style is validated when its components, such as pattern and construction detail are finalized. There is no need for any further validation at the production planning stage because the planning function's access is restricted to validated styles only.

7.2.3 Generalization of the Model

In the AS IS model, the sewing and finishing functions, which transform cut fabric into finished garment were modeled to reflect operations involved in the production of a particular type of garment. Each process step has been modeled as a sewing or finishing function as shown in Figure 7.7. In the TO BE function model, these functions were redefined to make the model a representative model for producing any type of garment. The generalization of the function model has been achieved by separating the generic functional

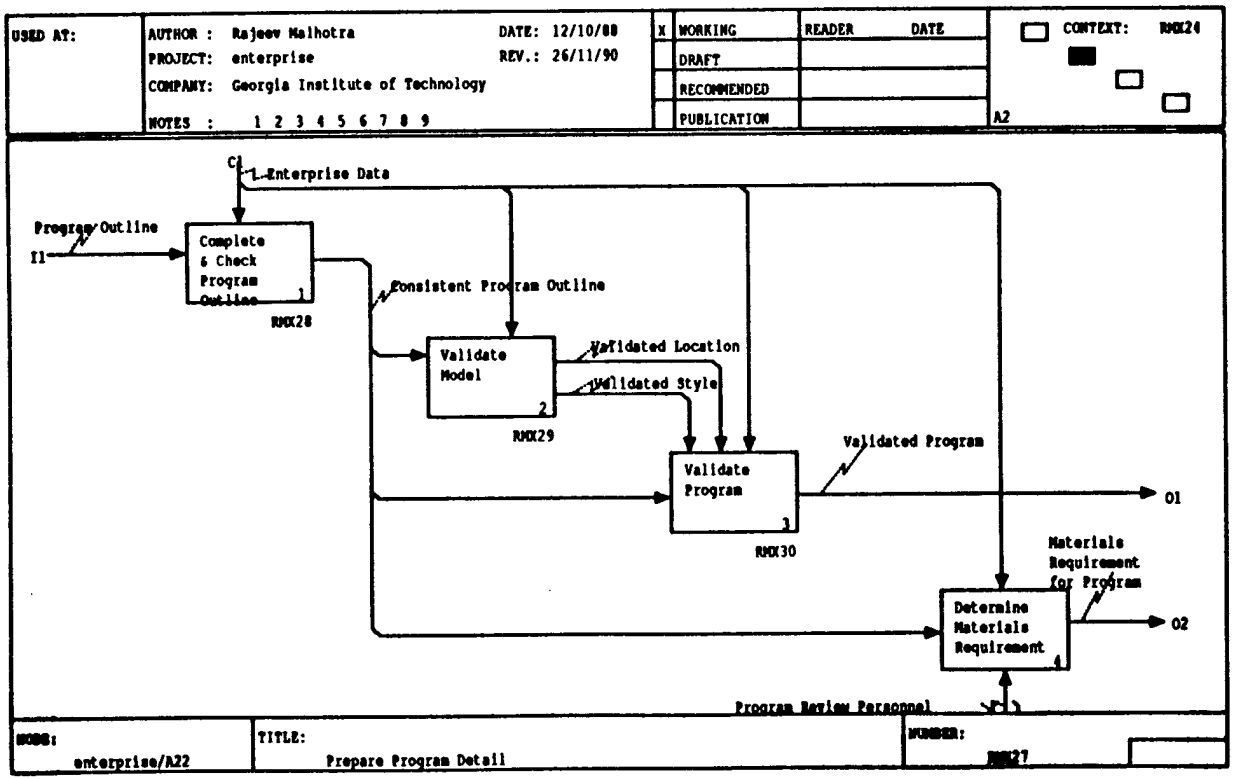


Figure 7.6. Representation of the *Validate Model* function in the AS IS function model

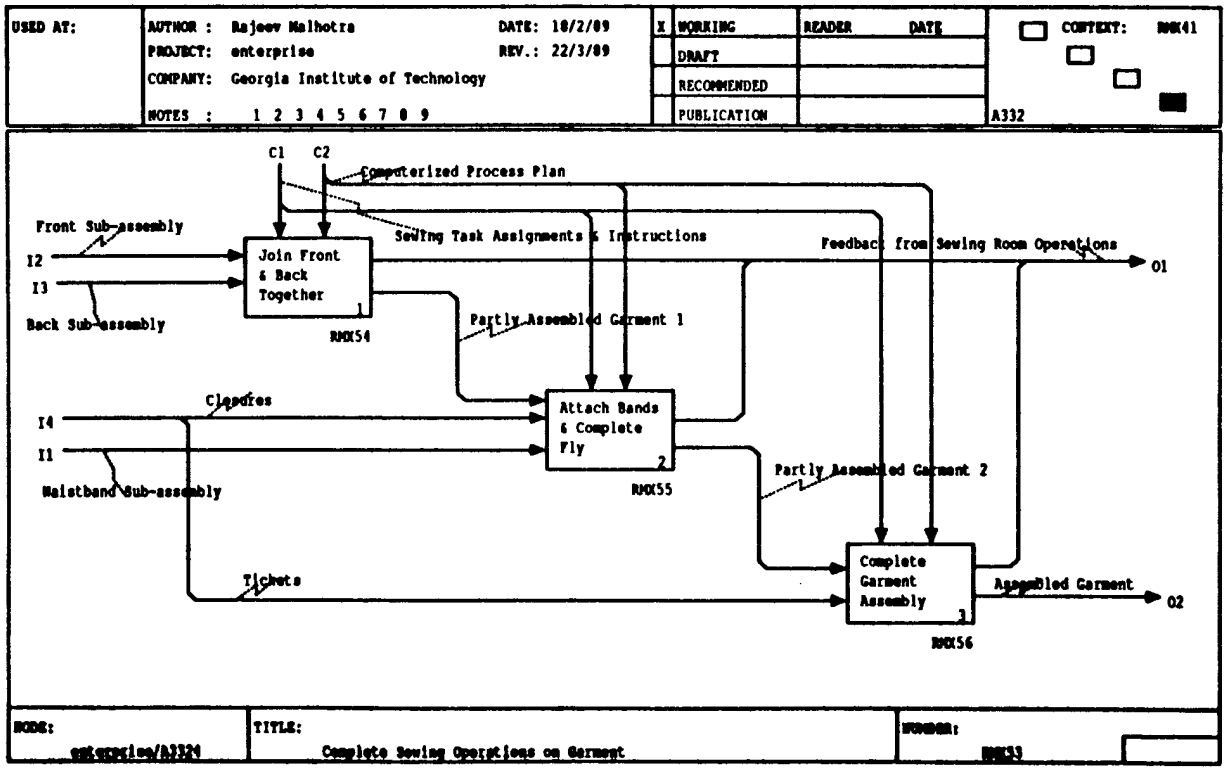


Figure 7.7. Representation of sewing operations in the AS IS function model

components of sewing and finishing from the process plan element that is specific to a particular garment. In the TO BE model the process plan has been abstracted as data which is represented by the ICOMs in the model. The TO BE function structure only contains the generic functions that are controlled by the process plans.

The generic apparel assembly functions have been modeled in diagram *Sew and Finish Garments* (A532) in the TO BE function model (Figure 7.8). In this model, sewing and finishing operations are performed on garments through four generic functions. The function *Hold Garment Sub-Assemblies* (A5322) represents the buffering function on the shopfloor. The in-process sub-assemblies are stored in buffers between operations. The function *Transport Garment Sub-Assemblies* (A5323) represents movement of buffered sub-assemblies to the *Process Garment Sub-Assemblies* (A5324) function which performs the actual sewing or finishing operations on the garment sub-assembly. The hold, transport and process functions are controlled by a feedback control function *Control Sewing and Finishing Units* (A5321) which controls a processing line or modules. The control function is driven by the process plan data which is part of the data represented by the ICOM *Assignments - Plant Resources* that interfaces to this function. Depending on the process plan, the control function can send garment sub-assemblies through a particular sequence of holds, transports and processes to yield garments of a particular style. Thus, the function *Sew and Finish Garments* represents a garment manufacturing system that is flexible and data driven.

7.3 Role of Function Model in TO BE Architecture

The TO BE function model complements the TO BE information model as the design specification for a CIM system for apparel manufacturing. Whereas the information model provides a database schema for the information system that supports CIM in an apparel enterprise, the function model specifies how the components of the information sys-

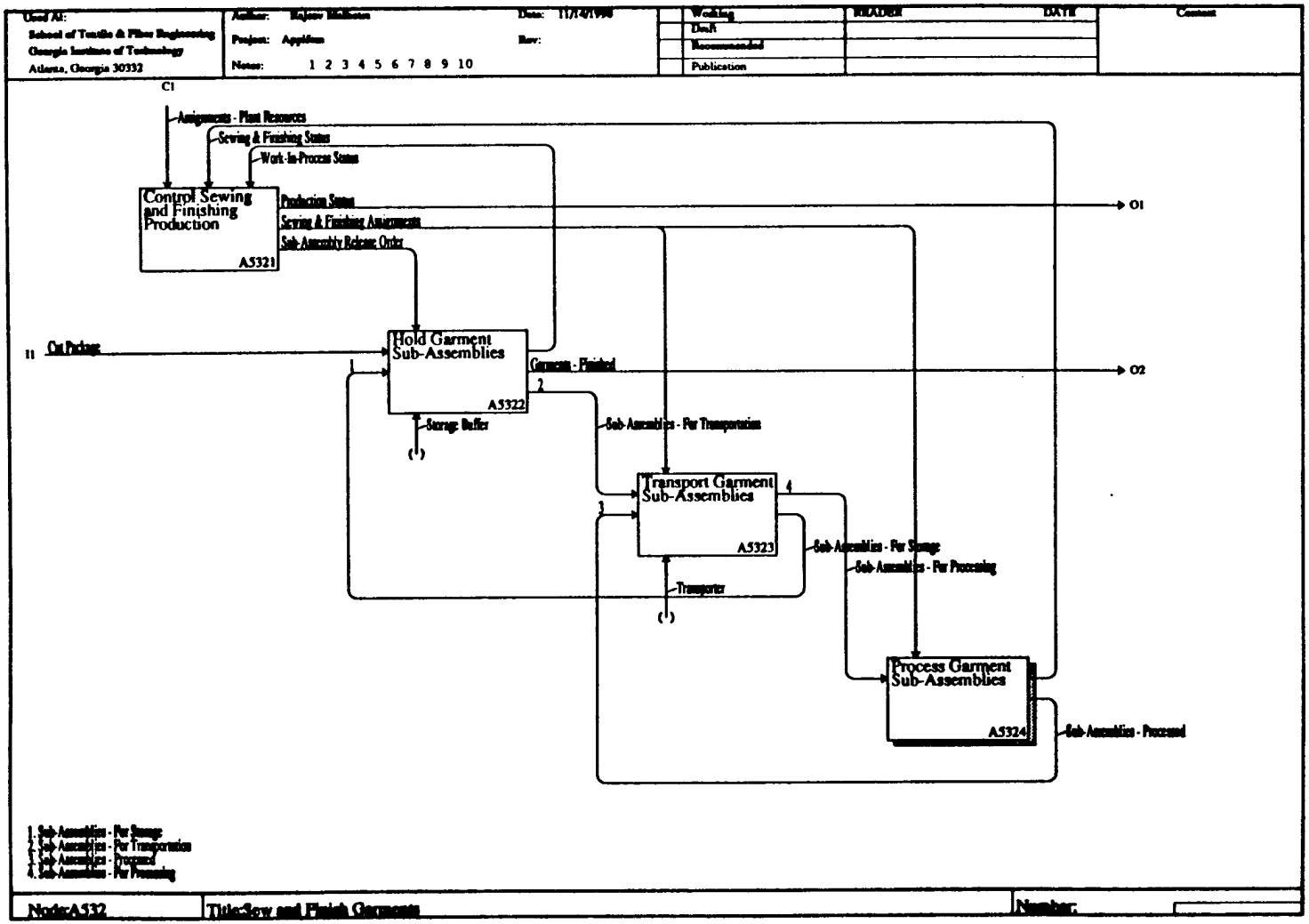


Figure 7.8. Representation of sewing and finishing functions in the TO BE function model

tem are distributed throughout the enterprise.

CHAPTER VIII

TO-BE ARCHITECTURE: THE DYNAMICS MODEL

The TO BE function and information models represent the static architecture of the proposed CIM system for an apparel manufacturing enterprise. However, to completely describe how the system functions, the time-varying behavior of the system needs to be modeled. The dynamics model of the system describes how the functions modeled in the function model and the information entities modeled in the information model interact dynamically to produce the desired system behavior. Thus, the TO BE dynamics model presented in this chapter provides the means for analyzing the behavior of the apparel enterprise.

8.1 Model Syntax and Conventions

The TO BE dynamics model was developed using the IFEM methodology discussed in Chapter V. In the IFEM methodology, the dynamics model is an extension of the function model. The TO BE dynamics model consists of *scripts* that describe how each lowest level function in the function model hierarchy behaves and interacts with other functions. The scripts capture the temporal interactions between the inputs, outputs, controls and mechanisms of a function. The sequence of dynamic actions performed by a function to transform its *input* and *control* entities into its *output* entities are expressed in a script for the function. The *mechanism* entities represent the resources which must be available for the transformations to take place. Examples of dynamic actions include engagement and release of resources, retrieval of entities from input queues, release of transformed entities into output queues, etc. From the viewpoint of the IFEM dynamics model, an ICOM inter-

face represents a channel along which entities can be moved between functions interconnected through the interface. The entities awaiting processing can queue up in the ICOM channel. Before the IFEM dynamics model is developed, the structure of entities corresponding to each ICOM has to be defined. In IFEM, the entity structure definition for each ICOM is derived from the information model as discussed in Chapter VII.

8.1.1 IFEM Dynamics Primitives

The dynamic actions modeled in the scripts are represented by a set of primitives. In general, each primitive has the following syntax:

Operation ICOM [*Selection Criteria*] [*Delay*];

Operation is the dynamic action that the primitive represents and *ICOM* is the interface channel on which the action is performed. The entities affected by the action are selected from the *ICOM* interface based on the specified *Selection Criteria*. *Delay* is the time lapse associated with the action. Consider the following example:

RETRIEVE I1 [I1.Color IS 'RED'][D1];

This primitive represents retrieval of all entities, whose feature *Color* has value *'RED'*, from input interface I1. The time taken to complete this action is D1. The entities selected as a result of this action are assigned to a collection named I1. The IFEM dynamics modeling primitives are listed in Table 1.

Engagement of Resources: Resources are engaged by functions using the ENGAGE primitive which has the following syntax:

ENGAGE M*x* [*selection criteria*];

The resources represented by the mechanism interface arrow M*x* are engaged. The *selection criteria* can be used to select resources based on their specific attributes. If the selection criteria are not specified, the first available resource entity is selected. The engaged resources are assigned to a collection named M*x* as a result of ENGAGE. If the requested re-

ENGAGE M*x* [*selection criteria*];	Engage resources from interface Mx to carry out a process
DISENGAGE M*x* [*selection criteria*];	Free up resources previously engaged from interface Mx
RETRIEVE I*x* (or C*x*) [*selection criteria*] [*delay*];	Retrieve entities from interface Ix or Cx for processing
COLLECT I*x* (or C*x*) [*selection criteria*] [*delay*];	Retrieve entities and add them to a set of previously retrieved entities
ASSIGN O*x.Feature.* <- *value* [*selection criteria*] [*delay*];	Assign a value to the specified feature of an entity associated with interface Ox
Process O*x* [*selection criteria*] [*delay*];	User-defined primitive for operating on entities associated with interface Ox
RELEASE O*x* [*selection criteria*] [*delay*];	Release processed entities at interface Ox
RELEASE O*x.Feature* <- *value* [*selection criteria*] [*delay*];	Release entities after assigning a value to the specified feature
LOOKUP I*x* (or C*x*, O*x*, M*x*) [*selection criteria*] [*delay*];	Lookup data representing entities associated with an ICOM interface
WAIT [*duration* / UNTIL *time*];	Wait for a specified duration or until a specified time
TRIGGER *condition*;	Activate a function when the specified condition is met
END;	Terminate a function
IF [*condition*] {*block 1*} ELSE {*block 2*};	Choose between alternate sequences of actions on the basis of the specified condition
WITH *p* {*block 1*} ELSE {*block 2*};	Choose between alternate sequences of actions depending on probability *p*
REPEAT [*control*] {*block*};	Repeat a sequence of actions
Control:	
[*n* TIMES]	Repeat *n* times;
[EACH *X* IN *collection*]	Repeat for each item *X* in the *collection*;
[WHILE *condition*]	Repeat while *condition* holds true;
[UNTIL *condition*]	Repeat until *condition* becomes true;

Table 1. The IFEM Dynamics Modeling Primitives

sources are busy, ENGAGE waits till they become available. However, resources may be preempted from other functions by using the PRIORITY(n) clause in the selection criteria. A higher value of *n* in the PRIORITY clause signifies a higher priority. A higher priority ENGAGE can interrupt an ongoing process with a lower priority ENGAGE. By default, the ENGAGE primitive has the lowest priority value of zero.

Release of Resources: To represent the release of engaged resources, once a task is finished, the following primitive is used:

DISENGAGE M*x* [*selection criteria*];

DISENGAGE releases the collection of resources previously engaged using the ENGAGE primitive and assigned to collection M*x*. The *selection criteria* can be used to release only those resources that meet the specified conditions. After a DISENGAGE, the resources that meet the selection criteria are removed from the collection M*x*. If a selection criterion is not specified, then by default, all the resources contained in the collection M*x* are released.

Retrieval of Entities: Entities queued up at the inputs or control interfaces of a function can be retrieved for processing using the RETRIEVE or COLLECT primitives, whose syntax is as follows:

RETRIEVE I*x* (or C*x*) [*selection criteria*] [*delay*];

COLLECT I*x* (or C*x*) [*selection criteria*] [*delay*];

Both RETRIEVE and COLLECT primitives remove entities that are queued at the specified input interface I*x* (or control interface C*x*) and meet the *selection criteria*, and assigns them to a collection named I*x* (or C*x*). The difference between RETRIEVE and COLLECT is as follows: when entities are retrieved repeatedly, RETRIEVE replaces the collection of previously retrieved entities by the newly retrieved ones; with COLLECT, the newly retrieved entities are added to the collection of previously retrieved entities.

If no selection criterion is provided, by default, the first entity in the queue is retrieved. If the specified queue does not contain the entities that meet the selection criterion

the RETRIEVE and COLLECT primitives wait till the entities become available, before retrieving them.

Processing of Entities: Processing of entities by functions involves modification of their attributes. For example a function may process a *job* entity by changing its *status* attribute from *in-process* to *processed.* The following primitives are used for representing transformation of entities by the functions:

ASSIGN O*x*.*Feature*. <- *value* [*selection criteria*] [*delay*];

Process O*x* [*selection criteria*] [*delay*];

The ASSIGN primitive is used to assign a value to a feature of the specified entity O*x*. All the entities that meet the *selection criteria* are selected for processing by the ASSIGN primitive and are collected in O*x*. In IFEM, processing is permitted only on the output entities.

A user-defined **Process** module is used to represent a complex set of actions, such as an update of a schedule, revision of a plan, etc., that cannot be represented using the ASSIGN primitive. Typically, the modifications represented by a process module require human skills or complex software. Each process module is defined in the script in which it is used. The definition only states what the module does; the design and implementation of the module is left to the system development phases that follow architecture development.

Release of Entities: The entities transformed by a function are released and queued up for other functions using the QUEUE primitive which has the following syntax:

QUEUE O*x* [*selection criteria*] [*delay*];

QUEUE selects entities that meet the *selection criteria* and puts them in the queue associated with the output interface O*x*. The selected entities are assigned to the collection named O*x* before they are queued. If no selection criterion is specified, the entities previously contained in the collection O*x* are queued. The QUEUE primitive also generates a signal to mark its completion. This signal is used by the function, from which the signal originates, to notify other functions of the queuing event.

An ASSIGN followed by a QUEUE can be combined into a single primitive as follows:

QUEUE O*x*.*Feature* <- *value* [*selection criteria*] [*delay*];

The specified *feature* of entities selected for queuing are assigned the *value* before queuing.

Data Lookup: All the entities that correspond to the structure definition of an ICOM interface can be looked up using the following primitive:

LOOKUP L*x* (or C*x*, O*x*, M*x*) [*selection criteria*] [*delay*];

The LOOKUP primitive searches for entities associated with an ICOM interface and meeting the *selection criteria* from the database of entities and assigns the selected entities to a collection. Unlike RETRIEVE, the LOOKUP primitive does not confine its search to the entities queued up at the interface, but searches from all the entities matching the interface definition. If no selection criterion is specified, all the entities that match the ICOM interface definition are selected.

Waiting Period between Actions: A waiting period between actions is modeled using the WAIT primitive which has the following syntax:

WAIT [*duration* / UNTIL *time*];

The *duration* or the *time* until which the WAIT has to remain in effect may be specified.

Conditional Branching: Conditional branching is modeled in the dynamic model using the following primitives:

IF [*condition*] {*block 1*} ELSE {*block 2*};

WITH *p* {*block 1*} ELSE {*block 2*};

The IF primitive is used for conditional branching in the scripts. If the condition holds true, *block 1* is executed; otherwise *block 2* is executed. A block contains one or more primitives representing a sequence of actions. Probabilistic branching is modeled using the WITH primitive. The probability that *block1* will be executed is *p*; *block 2* is executed if *block 1* is not. The ELSE clause is optional in the IF and WITH primitives.

Loops: Repeated execution of a sequence of actions is modeled using the REPEAT primitive which has the following syntax:

REPEAT [*control*] {*block*};

The *block* of actions is executed repeatedly and the number of repetitions is controlled by the *control* clause, which can be one of the following:

[*n* TIMES] : Repeat *n* times;

[EACH *collection*] : Repeat for each item in the *collection*;

[WHILE *condition*] : Repeat while *condition* holds true;

[UNTIL *condition*] : Repeat until *condition* becomes true;

Activation of Functions: The events that activate a function are specified using the TRIGGER primitive which has the following syntax:

TRIGGER *condition*;

A function is triggered by a set of events specified in the *condition* expression. There are three types of events that can be included in the *condition*: input, resource and clock events. The input event occurs when entities are queued at an input or control interface through the QUEUE primitive. The resource event occurs when a resource becomes available. The clock events are chronological events that occur when the clock reaches a preset time. The input and resource events are identified by the ICOM codes of the respective interfaces. The use of TRIGGER primitive is illustrated by the following example:

TRIGGER (I1 AND M2) OR NOON;

In this example, if NOON has been assigned a time value 12:00:00, the condition states that the function is activated either when input I1 and resource M2 are available, or at noon every day.

Termination of Functions: The termination of a function is indicated by the END primitive. If any resources are still engaged at the time the END primitive is encountered, these resources are released before the function is terminated.

8.1.2 The Selection Criteria

The entities processed by a primitive are selected by applying the specified selection criterion to the collection of candidate entities from which the selection is made. The selection criteria may consist of a set of individual criteria separated from each by commas. The syntax of the selection criteria is as follows:

[*Criterion1*, *Criterion2*,.....]

The criteria are applied to the candidate collection from left to right, i.e., *criterion2* is applied to the collection of entities selected after applying *criterion1*.

An individual criterion may be a boolean selection condition, that must be satisfied by the selected entities. A selection condition consists of relational expressions, that match a feature of an entity to a value, connected together by logical operators AND and OR. For example, the condition that the entities selected from input I1 have been created by person identified as M1 and are not red in color is specified as:

I1.Color NOT 'RED' AND I1.Creator IS M1

The following relational operators can be used in the selection conditions:

IS : Is same as;

NOT : Is different from;

IN : Is contained in;

NOT IN : Is not contained in;

The IS operator checks whether the items being compared are the same. The NOT operator is the inverse of IS. The items compared using the IS and NOT operators can be entities, features or collections of entities. When collections are compared, then the condition holds true if the collections being compared contain exactly the same items. The IN operator checks whether, in the relational expression A IN B, all the entities in collection A are contained in collection B. Relational expressions involving numerical or enumerated items can also use the GT (greater than), GE (greater than or equal to), LT (less than) and LE (less

than or equal to) relational operators.

A selection criterion may also be a function that selects and returns a collection of entities. For example, the function FIRST(*N*) returns the first *N* entities from the collection of candidate entities. The following functions can be used in selection criteria:

FIRST(N)	: Select first N entities from the collection;
LAST(N)	: Select last N entities from the collection;
SAME(feature)	: The specified feature has the same value for all the selected entities;

Other selection functions may be defined to suit the modeling needs. In defining a selection function, only a description of what the function does is provided; how the function is implemented is not part of the IFEM dynamics model. User-defined selection functions may represent selection criteria that require human skills or considerably intricate selection logic.

8.2 The TO BE Dynamics Model

The primitives discussed in the previous section were used to develop scripts that model the dynamics of each low-level function contained in the TO BE function model. To illustrate how the dynamics of the TO BE apparel enterprise are represented in the model, the dynamics scripts of functions involved in converting cut fabric parts into finished garments, represented by the node *Sew and Finish Garments* (A532), are discussed in detail. Appendix C contains the complete TO BE dynamics model.

8.2.1 An Overview of the Structure of the *Sew and Finish Garments* Function

The *Sew and Finish Garments* function models the activities of the sewing and finishing shopfloor (Figure 8.1). The input to this function is *Cut Package Shipment* which represents all the cut fabric parts and construction materials required to produce garments

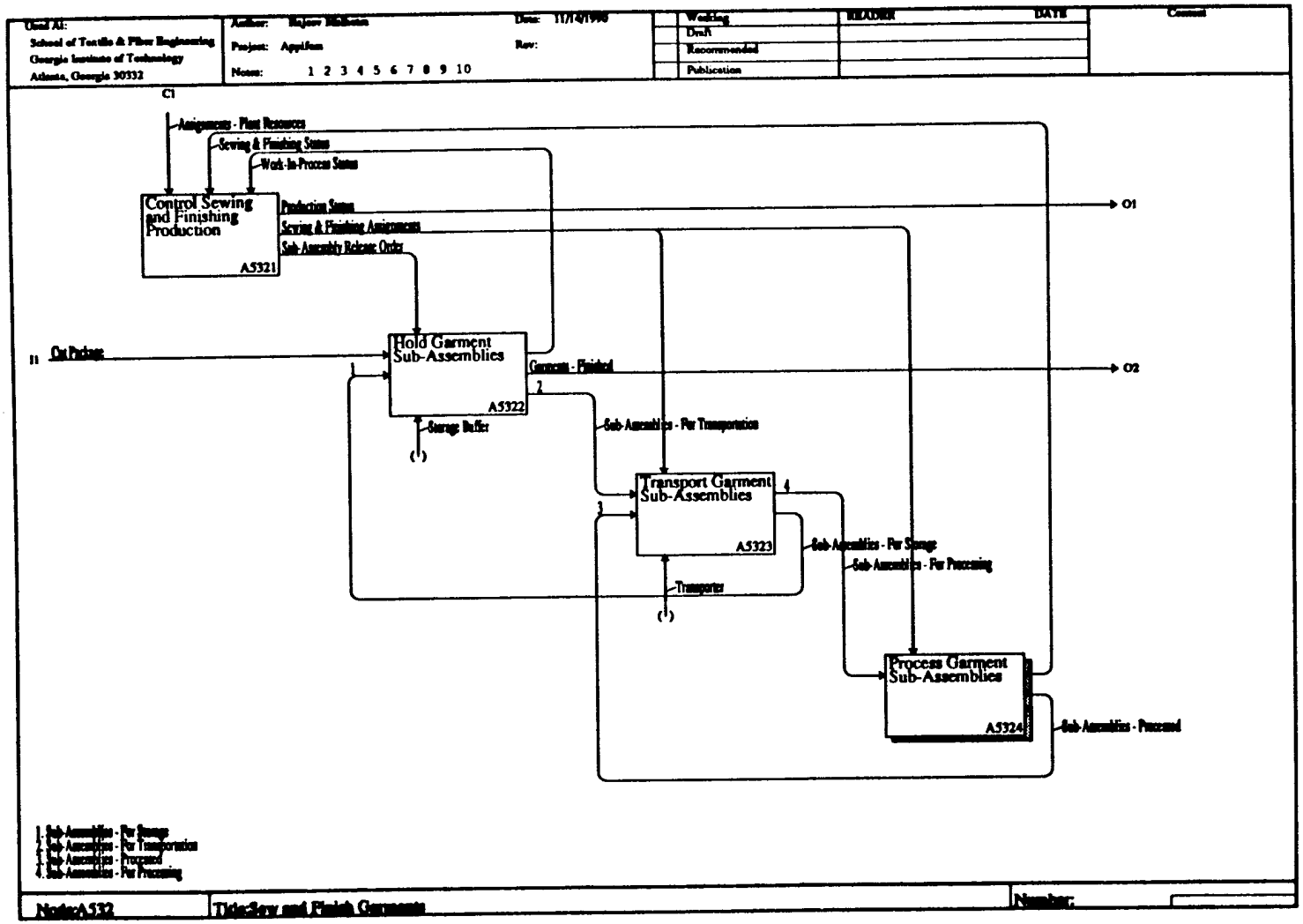


Figure 8.1. Breakdown of the *Sew and Finish Garments* function

for an order. The control interface to this function is *Assignments - Plant Resources* that represents the resources assigned for performing each step involved in producing garments for the order. The resources assigned to produce garments are grouped into functional modules, each containing one or more workstations and designated to perform a set of operations. For example, six modules may be assigned for an order - one each for front, back, waistband and final assembly, and two for finishing. One or more operators is assigned to operate workstations in each module. The outputs from the function are finished garments and production status information represented by output interfaces *Garments - Finished* and *Production Status* respectively.

The *Sew and Finish Garments* function is detailed further to represent the basic shopfloor activities. The production operations are carried out at the sewing and finishing modules assigned to the order. Garment sub-assemblies for an order are held in a storage buffer from where they are taken to the modules for processing and brought back by transport devices serving the modules. The movement of sub-assemblies to the modules is controlled by the shopfloor controller that routes the sub-assemblies to the appropriate modules and ensures the correct sequencing of operation.

8.2.2 Dynamics of the *Sew and Finish Garment* Function

The script corresponding to each lowest-level function under the *Sew and Finish Garments* node describes how that function is activated and how it behaves once it is activated. The structure definitions of the entities available to each function are contained in the TO BE function model glossary.

Control Sewing and Finishing Production: The shopfloor control activities are represented by the function *Control Sewing and Finishing Production* (A5321). The script describing the dynamics of this function is shown in Figure 8.2. This function is activated when the entity *Assignments - Plant Resources* is available at the control interface C1. This

```
Control Sewing and Finishing Production
Description:
    Control the real-time operation of sewing/finishing lines or modules. Move each
garment unit through the entire sequence of process steps and track its status all
along. Also ensure that the process steps are performed in the correct sequence given
on the process plan.
Interface:
    C1: Assignments - Plant Resources;        (F77{SCH_PROD_ORD})
    C2: Sewing & Finishing Status;            (F78{PROD_ASSGNMT})
    C3: Work-In-Process Status;               (F101{GARSUBASSEM})
    O1: Production Status;                    (F48{PRODUCTION_ORDER})
    O2: Sewing & Finishing Assignments;       (F78{PROD_ASSGNMT})
    O3: Sub-Assembly Release Order;           (F101{GAR_SUBASSEM})
Dynamics:
    Select1 :[Selects sub-assemblies for further processing];
    Proc1   :[Assigns the next processing location to selected sub-as];
    TRIGGER : C1;
    RETRIEVE C1;
    REPEAT [EACH C1.Assgnmt]
        IF [C1.Assgnmt.Equip.Function IN ('SEWING', 'FINISHING')]
            QUEUE O2 [O2 IS C1.Assgnmt];
    REPEAT [UNTIL C1.Order.Status IS 'FINISHED']
        (
        RETRIEVE C3 [C3.Unit.Order IS C1.Order AND Select1];
        Proc1 O3 [O3 IS C3];
        QUEUE O3;
        LOOKUP C2 [C2 IN C1.Assgnmt AND C2.Status NOT 'DONE' AND
                   C1.Equip.Function IN ('SEWING', 'FINISHING')];
        IF [C2 IS NULL]
            QUEUE O1.Status <- 'FINISHED' [O1 IS C1.Order];
        )
    END;
```

Figure 8.2. Dynamics description script for the *Control Sewing and Finishing Production* function

entity is retrieved from C1 by the RETRIEVE primitive. The retrieved entity contains a list of assignments *Assgnmt* for individual modules to be used for producing garments in the order. The assignments for modules (*Equip*) with function 'SEWING' or 'FINISHING', are queued at the output interface O2.

The sequence of actions inside the REPEAT loop is repeated until the status of the entire order is changed to 'FINISHED'. From C3, the sub-assemblies that belong to the order being processed and selected by the *Select1* module for further processing, are retrieved. The entities retrieved from C3 are processed by the *Proc1* module and released at the interface O3 by the QUEUE primitive. Next, the sewing and finishing assignments for this order, whose status is not 'DONE' are looked up from the interface C2. If C2 does not contain any assignment with status not 'DONE', i.e., all the assignments are finished, the entity *Production Status* is released at the interface O1 after updating its status to 'FINISHED'. Once the REPEAT loop is exited, the function activation is terminated by the END primitive.

The selection function *Select1* encapsulates the logic used by the controller to select sub-assemblies for further processing. The user-defined procedure *Proc1* encapsulates the steps for assigning the next processing location to which the selected sub-assemblies have to be routed. To reiterate, in IFEM dynamics models, modules such as *Select1* and *Proc1*, are only declared; the detailed design and implementation of these modules follows architecture development.

Hold Garment Sub-Assemblies: The function *Hold Garment Sub-Assemblies* (A5322) represents buffering of in-process sub-assemblies. The script describing the dynamics of this function is shown in Figure 8.3. This function is activated by the arrival of the entity *Cut Package* at the input interface I1. A storage buffer is engaged from interface M1. A cut package is retrieved from input I1. All the sub-assemblies contained in the entity retrieved from I1 are released as work-in-process status at the output O1 (this output pro-

```
Hold Garment Sub-Assemblies
Description:
    Hold the garments sub-assemblies in process between process steps. Update the lo-
cation of each garment sub-assembly received in the buffer.
Interface:
    I1: Cut Package;                          (F48{PRODUCTION_ORDER})
    I2: Sub-Assemblies - For Storage;         (F101{GAR_SUBASSEM})
    C1: Sub-Assembly Release Order;           (F101{GAR_SUBASSEM})
    O1: Work-In-Process Status;               (F101{GAR_SUBASSEM})
    O2: Garment - Finished;                   (F79{GARMENT_UNIT})
    O3: Sub-Assemblies - For Transportation;  (F101{GAR_SUBASSEM})
    M1: Storage Buffer;                       (F30{BUFFER})
Dynamics:
    TRIGGER :I1;
    ENGAGE M1;
    RETRIEVE I1;
    QUEUE O1 [O1 IN I1.Unit.SubAssem];
    REPEAT [UNTIL I1.Status IS 'FINISHED']
        {
        RETRIEVE C1 [C1.Unit.Order IS I1];
        QUEUE O3 [O3 IN C1];
        RETRIEVE I2 [I2.Unit.Order IS I1];
        QUEUE O1 [O1 IS I2];
        }
    QUEUE O2 [O2 IN I1.Unit];
    DISENGAGE M1;
    END;
```

Figure 8.3. Dynamics Description Script for the *Hold Garment Sub-Assemblies* Function

vides the *Control Sewing and Finishing Production* function with a list of sub-assemblies available in the buffer).

The sequence of actions within the REPEAT loop is repeated until the status of the order for which the cut package was retrieved from I1 becomes 'FINISHED'. The sub-assemblies marked for further processing by the *Control Sewing and Finishing Production* function are retrieved by the RETRIEVE primitive. The retrieved sub-assemblies are released for transportation to the manufacturing modules at the output interface O3. The processed sub-assemblies transported back from the manufacturing modules are retrieved from I2. These sub-assemblies are released for the *Control Sewing and Finishing Production* function at O1 by the QUEUE primitive.

When the REPEAT loop is exited, the finished garments are released at the interface O2 by the QUEUE primitive. Next, the storage buffer engaged from M1 is released and the function activation is terminated.

Transport Garment Sub-Assemblies: The movement of sub-assemblies between the modules and storage is represented by the function *Transport Garment Sub-Assemblies* (A5323). The script describing the dynamics of this function is shown in Figure 8.4. This function is activated when the entity *Sewing & Finishing Assignments* is available at the interface C1. This entity is retrieved from C1.

The sequence of actions within the REPEAT loop is repeated until the status of the entity retrieved from C1 becomes 'DONE'. A transporter resource, e.g., a trolley or a conveyor, that belongs to the manufacturing module to be used for the assignment retrieved from C1 is engaged. The sub-assemblies, released for further processing by the *Hold Garment Sub-Assemblies* function, and routed to the manufacturing module served by this transporter are retrieved from I1 and released at the interface O1 for processing at the module. The sub-assemblies processed by the module are retrieved from I2. The *Loc* attribute of these sub-assemblies is assigned the value NULL before releasing at O2 for the *Hold*

```
Transport Garment Sub-Assemblies
Description:
    Move garment sub-assemblies between storage buffer and processing units.
Interface:
    I1: Sub-Assemblies - For Transportation;  (F101{GAR_SUBASSEM})
    I2: Sub-Assemblies - Processed;           (F101{GAR_SUBASSEM})
    C1: Sewing & Finishing Assignments;       (F78{PROD_ASSGNMT})
    O1: Sub-Assemblies - For Processing;      (F101{GAR_SUBASSEM})
    O2: Sub-Assemblies - To Buffer;           (F101{GAR_SUBASSEM})
    M1: Transporter;                          (F72{TRANSPORTER})
Dynamics:
    D1          :[Time to transport];
    TRIGGER     :C1;
    RETRIEVE C1;
    REPEAT [UNTIL C1.Status IS 'DONE']
        {
        ENGAGE M1 [M1.Group IS C1.Equip];
        #Transport from storage to processing unit
        RETRIEVE I1 [I1.Loc IS C1.Equip];
        QUEUE O1 [O1 IS I1] [D1];
        #Transport from processing unit to storage
        RETRIEVE I2 [I2.Loc IS C1.Equip];
        QUEUE O2.Loc <- NULL [O2 IS I2] [D1];
        DISENGAGE M1;
        }
    END;
```

Figure 8.4. Dynamics description script of the *Transport Garment Sub-Assemblies* function

Garment Sub-Assemblies function. The transporter resource engaged from M1 is released. When the REPEAT block is exited, the function is terminated by the END primitive.

Process Garment Sub-Assemblies: The activities of a manufacturing module are modeled as the function *Process Garment Sub-Assemblies* (A5324) which is broken down further in diagram node A5324. The dynamics of this function is described by the scripts developed for its sub-functions. This function represents a manufacturing module that performs the process steps assigned to it on sub-assemblies sent to it and returns them as *Sub-assemblies - Processed*. When the assigned work is completed the status of the work assignment is updated to 'DONE'.

8.2.3 Dynamic Interaction Between Functions

In the scripts describing the dynamics of individual functions, the dynamic interactions between the functions are implicitly modeled through the QUEUE and RETRIEVE primitives. In IFEM, ICOMs are viewed as channels along which entities flow between functions. The QUEUE primitive puts entities in this channel and the RETRIEVE primitive removes them from it. A RETRIEVE waits until the entities meeting the specified selection criterion for retrieval become available at the interface. For example, the *Control Sewing and Finishing Production* function waits for the control *Work-In-Process Status* to be made available by the *Hold Garment Sub-Assemblies* function before it proceeds with the actions that generate the output *Sub-Assembly Release Order*. The *Hold Garment Sub-Assemblies* function, in turn, queues the garment sub-assemblies at its output *Sub-Assemblies - For Transportation* only after a release order is available for retrieval at the control interface *Sub-Assembly Release Order*. Thus, the dynamics of interactions between functions is captured in the scripts through the use of the RETRIEVE and QUEUE primitives.

8.3 Role of Dynamics Model in the TO BE Architecture

The TO BE dynamics model complements the static architecture of a CIM system for an apparel enterprise, as described by the TO BE function and information models, in providing a comprehensive description of how the system works. The process of developing the dynamics model can be viewed as the final stage in developing the specification for the TO BE enterprise. In the dynamics description scripts of each function, the modules that must be designed and implemented to perform specific tasks associated with each function are identified. These modules perform two types of actions: *control* and *processing*. The *control* modules are identified in the scripts as selection functions used to specify complex selection criteria associated with various dynamic actions. The *process* modules perform actions that transform an entity. For example, in the *Control Sewing and Finishing Production* function (Figure 8.2), the selection function *Select1* is a *control* module whereas the procedure *Proc1* is a *process* module.

The TO BE dynamics model also provides a context for setting up a simulation of the TO BE enterprise through which the design of the proposed system could be evaluated and its performance can be analyzed prior to its implementation. However, to simulate the system, the model must be supplemented with *enterprise-specific* data such as processing delays, arrival rates for orders and production capacities.

CHAPTER IX

CONCLUSIONS AND RECOMMENDATIONS

It has been established that an apparel manufacturing enterprise can benefit from the flexibility and opportunities for quick response offered by adopting the CIM approach. The need for a systematic approach to design and planning of CIM systems has been identified as critical to the successful implementation of CIM in an enterprise. The first step in the systematic approach is development of an architecture of the enterprise which provides an understanding of how the enterprise functions as it exists and how it would function under the proposed CIM system.

9.1 Conclusions

The research work presented here covers two areas: apparel manufacturing and manufacturing systems modeling. An architecture consisting of a set of models that provides specifications of a CIM system for an apparel manufacturing enterprise has been developed. A methodology suited to the specific needs of CIM system modeling has been presented and used to develop the architecture for an integrated apparel enterprise.

9.1.1 Architecture for Computer-Integrated Manufacturing of Apparel

The basis of the apparel manufacturing architecture is the AS IS function model, presented in Chapter IV, which represents the functions performed by an existing apparel manufacturing enterprise and the relationships that exist between these functions. The AS IS function model provides an understanding of the apparel manufacturing domain based on which the TO BE architecture has been developed.

Central to CIM is an enterprise-wide information system, that supports

interactions between enterprise functions through information sharing. The TO BE information model, presented in Chapter VI, provides a comprehensive definition of the enterprise data suitable for information sharing and serves as the conceptual schema for a CIM information system for an apparel manufacturing enterprise.

The TO BE function model, presented in Chapter VII, provides the function structure of an apparel enterprise in which the functions interact with each other through the enterprise-wide CIM information system as defined by the TO BE information model. In this model, the inputs, controls, outputs and mechanisms for each function have been defined as interfaces to the CIM information system. The functions are viewed as nodes of the distributed information system. The functions maintain the data that they work with locally; the data that a function makes accessible to other functions for sharing is explicitly represented in the model through its output interfaces.

The time-varying behavior of the apparel enterprise, whose static structure is modeled in the TO BE function and information models, is captured in the TO BE dynamics model presented in Chapter VIII. This model describes the dynamic behavior of each individual enterprise function in terms of interactions among its inputs, controls, mechanisms and outputs, and the interactions between the functions through their interfaces. In the process of describing the dynamic behavior, the control and processing modules within each function are specified. The dynamics model also plays an important role in the detailed design phase by providing a context for simulation through which the system design can be tested prior to actual implementation.

Of the two aspects of CIM, i.e., information systemization and process mechanization, the TO BE architecture focuses mainly on the systemization aspect because systemization needs must be clearly identified at the enterprise level before mechanization for CIM at the function level can be designed and implemented. The mechanization aspect is addressed in the architecture by developing the specifications for the control and process

modules. How these tasks are mechanized and at what level mechanization is implemented is left to the detailed design and implementation phases.

9.1.2 The Modeling Methodology

The existing systems modeling methodologies, including the IDEF methodology, have been found deficient in many respects for the purpose of CIM system modeling. The available methodologies lack support for integrating the function, information and dynamics models of an enterprise into a comprehensive, yet consistent architecture. Without this support, it is very difficult to ensure that the function, information and dynamics models, do in fact, represent complementary aspects of the same system. This deficiency has been found particularly serious while modeling a considerably large and complex system such as an apparel manufacturing enterprise. As part of the proposed integrated framework for enterprise modeling (IFEM), a method for integrating the function, information and dynamics models into a single framework has been developed.

In dynamics modeling, the approach of modeling the flow of a single entity through the system, adopted by $IDEF_2$ and the general purpose simulation languages, has been found unsuitable for modeling CIM systems which need to reconfigure dynamically to concurrently process a variety of entities. In the dynamics modeling approach adopted in IFEM, the process sequence for a particular entity is not included in the dynamics model, but treated as a data input to it, thus facilitating the modeling of data-driven processes that make up a CIM system.

9.2 Recommendations

The concept of CIM is an evolutionary one and there is no single CIM solution for an apparel enterprise, let alone for an entire industry. Consequently, the TO BE architecture proposed here is not the ultimate architecture of a CIM system for apparel

manufacturing. It is only a starting point and there is scope for further refinement. Each functional area modeled in the TO BE architecture could be taken up for a more in-depth analysis. Future research efforts could also focus on the mechanization aspect of CIM for apparel enterprises. Design and development of modules for the control and processing tasks identified in the architecture, especially in the functional areas where human intelligence plays a critical role, constitute interesting problems which could be the subject of further investigation.

The proposed IFEM methodology provides a concise description of the dynamics of the system being modeled and could be a suitable basis for the design of a general-purpose simulation language. Since an IFEM dynamics model uses entity definitions from the associated information model, a simulation based on the IFEM model could be linked to the enterprise database and used in a decision support role. The possibility of using IFEM models in simulation-based decision support systems could also be investigated.

BIBLIOGRAPHY

Adiga88 Adiga, S., and Glassey, C.R., "Object-Oriented Simulation to Support Research in Manufacturing," Technical Report ESRC 88-20, University of California at Berkeley, Berkeley, CA, December 1988.

AIT88 Apparel Industry Trends, American Apparel Manufacturers Association, Arlington, VA, September 1988.

AMTC89 Jayaraman, S., and Malhotra, R., "Apparel Manufacturing Architecture: The Function Model," *AMTC Quarterly*, May 1989, pp. 1-9.

Banks88 Banks, J., and Carson, J. S., "Applying the Simulation Process," Proceeding of the 1988 Winter Simulation Conference (Abrams, M., Haigh, P., Comfort, J., eds.), Institute of Electrical and Electronic Engineers, San Diego, CA, 1988.

Bowers89 Bowers, D. S., "From database to information base: some questions of semantics and constraints," *Information and Software Technology*, Vol. 31, No. 8, October 1989, pp. 402-410.

Chen76 Chen, P. P. S., "The Entity-Relationship Model - Towards a Unified View of Data," *ACM Transactions on Database Systems*, Vol. 1, No. 1, March 1976, pp. 9-36.

Codd70 Codd, E. F., "A Relational Model of Data for Large Shared Data Banks," *Communications ACM*, Vol. 13, No. 6, June 1970, pp. 377-387.

Codd79 Codd, E. F., "Extending the Database Relational Model to Capture More Meaning," *ACM Transactions on Database Systems*, Vol.4, No. 4, december 1979, pp. 397-434.

Dadam89 Dadam, P. and Linnemann, V., "Advanced Information Management (AIM); Advanced database technology for integrated applications," *IBM Systems Journal*, Vol. 28, No. 4, 1989, pp. 661-681.

Davis88 Davis, D. A., and Pegden, C.D., "Introduction to SIMAN," Proceed-

ing of the 1988 Winter Simulation Conference (Abrams, M., Haigh, P., and Comfort, J., eds.), Institute of Electrical and Electronic Engineers, San Diego, CA, 1988.

DeMarco78 DeMarco, T., Structured Analysis and System Specification, Yourdon Press, New York, NY, 1978.

DeMarco79 DeMarco, T., Concise Notes on Software Engineering, Yourdon Press, New York, NY, 1979.

Gaylord87 Gaylord, J., Factory Information Systems - Design and Implementation for CIM Management and Control, Marcel Dekker Inc., New York, NY, 1987.

Garvey89 Garvey, M. A., and Jackson, M. S., "Introduction to object-oriented databases," Information and Software Technology, Vol. 31, No. 10, December 1989, pp. 521-528.

Godwin89 Godwin, A.N., Gleeson, J.W., and Gwillian, D., "An Assessment of the IDEF Notations as Descriptive Tools," *Information Systems*, Vol. 14, No. 1, 1989, pp.13-28.

Gordon78 Gordon, G., System Simulation, Prentice Hall Inc., Englewood Cliffs, NJ, 1978.

Hammer81 Hammer, M., and McLeod, D., "Database Description with SDM: A Semantic Data Model," ACM Transactions on Database Systems, Vol. 6, No. 3, September 1981.

ICAM81a Integrated Computer-aided manufacturing (ICAM), Function Modeling Manual $IDEF_0$, Materials Laboratory, Air Force Wright Aeronautical Laboratories, AFSC, Wright-Patterson AFB, OH, 1981.

ICAM81b Integrated Computer-aided manufacturing (ICAM), Dynamic Modeling Manual $IDEF_2$, Materials Laboratory, Air Force Wright Aeronautical Laboratories, AFSC, Wright-Patterson AFB, OH, 1981.

IISS85 Integrated Information Support System (IISS), Information Modeling Manual $IDEF_1x$, ICAM Project Priority 6201, D. Appleton Company Inc., Manhattan Beach, CA, 1985.

Jayaraman88a Jayaraman, S., "Design and Development of a Generic Architecture for Apparel Manufacturing," Georgia Institute of Technology Re-

search Proposal submitted to Defense Logistics Agency, Cameron Station, VA, April 1988.

Jayaraman90 Jayaraman, S., "Design and Development of an Architecture for Computer-Integrated Manufacturing in the Apparel Industry, Part I: Basic Concepts and Methodology Selection", *Textile Research Journal*, Vol. 60, No. 5, May 1990, pp. 247-254.

KBSI89 AI0 -- Computer-Aided IDEF0 Function Modeling System, Knoeledge Based Systems, Inc., College Station, Texas, 1989.

Lyngbaek86 Lyngbaek, P., and Kent, W., "A Data Modeling Methodology for the Design and Implementation of Information Systems," Proceedings of 1986 International Workshop on Object-Oriented Database Systems, Pacific Grove, CA, September 1986.

Mackulak84 Mackulak, G. T., "High Level Planning and Control: An $IDEF_0$ Analysis for Airframe Manufacture," *Journal of Manufacturing Systems*, Vol. 3, No. 2, 1984, pp. 121-132.

Malhotra90 Malhotra, R., and Jayaraman, S., "Design and Development of an Architecture for Computer-Integrated Manufacturing in Apparel Industry, Part II: The Function Model," *Textile Research Journal*, Vol. 60, No. 6, June 1990, pp. 351-360.

Miles88 Miles, T., Sadowski, R. P., and Werner, B. M., "Animation with CINEMA," Proceeding of the 1988 Winter Simulation Conference (Abrams, M., Haigh, P., and Comfort, J., eds.), Institute of Electrical and Electronic Engineers, San Diego, CA, 1988.

Oracle88 Oracle Relational Database Management System, v. 6.0, Oracle Corporation, Belmont, CA, 1988.

O'Reilly88 O'Reilly, J. J., and Lilegdon, W. R., "SLAM II Tutorial," Proceeding of the 1988 Winter Simulation Conference (Abrams, M., Haigh, P., and Comfort, J., eds.), Institute of Electrical and Electronic Engineers, San Diego, CA, 1988.

Pegden82 Pegden, C. D., Introduction to SIMAN, System Modeling Corporation, Sewckley, PA, 1982.

Perkinson84 Perkinson, R. C., Data Analysis - The Key to Data Base Design, QED Information Sciences Inc., Wellesley, MA, 1984.

Piltzecker85 Piltzecker, E. W., and Sapita, R. F., "Modeling Management Operation Systems," *Control Engineering*, June 1985, pp. 104-107.

Pruett90 Pruett, J.M. and Vasudev V.K., "MOSES: Manufacturing Organization Simulation and Evaluation System," Simulation, January 1990, pp. 37-45.

Ranky86 Ranky, P. G., Computer Integrated Manufacturing, Prentice Hall International, Englewood Cliffs, NJ, 1986.

Rowe87 Rowe, L.A., and Stonebraker, M.R., "The POSTGRES Data Model," Proceedings of the 13th VLDB Conference, Brighton, UK.

Russell88 Russell, E. C., "SIMSCRIPT II.5 and SIMGRAPHICS: A Tutorial," Proceeding of the 1988 Winter Simulation Conference (Abrams, M., Haigh, P., and Comfort, J., eds.), Institute of Electrical and Electronic Engineers, San Diego, CA, 1988.

Schrefl90 Schrelf, M., and Kappel, G., "Using an Object-Oriented Diagram Technique for the Design of Information Systems," Proceedings of International Conference on Dynamic Modeling of Information Systems, Noordwijkerhout, Austria, April 1990.

Schriber88 Schriber, T. J., "Perspectives on Simulation using GPSS," Proceeding of the 1988 Winter Simulation Conference (Abrams, M., Haigh, P., and Comfort, J., eds.), Institute of Electrical and Electronic Engineers, San Diego, CA, 1988.

Shipman81 Shipman, D.W., "The Functional Data Model and the Data Language DAPLEX," *ACM Transactions on Database Systems*, Vol. 6, March 1981, pp. 140-173.

Solinger80 Solinger, J., Apparel Manufacturing Handbook, Van Nostrand Reinhold Company, New York, NY, 1980.

Stonebraker87 Stonebraker, M.R., "The Design of the POSTGRES Storage System," Proceedings of the 13th VLDB Conference, Brighton, UK.

STRADIS88 STRADIS: Information Systems Development Methodology, Product Description Manual, McDonnel Douglas, St. Louis, MO, 1988.

Suri88 Suri, R., and Tomsicek, M., "Rapid Modeling Tools for Manufactur-

ing Simulation and Analysis," Proceeding of the 1988 Winter Simulation Conference (Abrams, M., Haigh, P., and Comfort, J., eds.), Institute of Electrical and Electronic Engineers, San Diego, CA, 1988.

TAC88 1988 Report of the Technical Advisory Committee, American Apparel Manufacturers Association, Arlington, VA, 1988.

Wilkinson90 Wilkinson, K., Lyngbaek, P., and Hasan, W., "The Iris Architecture and Implementation," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2, No. 1, March 1990, pp. 63-75.

Wizdom88 IDEFine-0 and IDEFine-1, User Manuals, Wizdom Systems Inc., Naperville, IL, 1988.

Yeomans85 Yeomans, R. W., Choudry, A., and ten Hagen, P. J. W., Design Rules for a CIM System, North Holland, Netherlands, 1985.

Young88 Young, R.E., Vesterager, J., Wichmann, K.E., and Heide, J., "Simulation Uses in CIM Development," *International Journal of Computer Integrated Manufacturing*, Vol. 1, No. 1, Jan-March 1988, pp. 50-54.